TESIS DE DOCTORADO

"DISEÑO DE REDES NEURONALES BASADO EN MODELOS PERCEPTUALES MULTINIVEL PARA SEGMENTACIÓN DE OBJETOS ESTÁTICOS Y DINÁMICOS EN SECUENCIAS DE VIDEO"

QUE PARA OBTENER EL GRADO DE DOCTORADO EN CIENCIAS EN INGENIERÍA ELECTRÓNICA

DESARROLLÓ:

JUAN ALBERTO RAMÍREZ QUINTANA

ASESOR:

Dr. MARIO IGNACIO CHACÓN MURGUÍA

Instituto Tecnológico de Chihuahua

División de Estudios de Posgrado e Investigación

CHIHUAHUA CHIH., SEPTIEMBRE 2014

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Chihuahua

"2014 Año de Octavio Paz"

Chihuahua, Chih., 18 de septiembre de 2014

C. JUAN ALBERTO RAMÍREZ QUINTANA
PRESENTE

Por este conducto le comunico que a propuesta del Jurado de Examen Doctoral, la División de Estudios Posgrado e Investigación le ha concedido la autorización para la impresión de tesis para obtener el grado de Doctor en Ciencias en Ingeniería Electrónica, cuyo título es:

"Diseño de redes neuronales basado en modelos perceptuales multinivel para segmentación de objetos estáticos y dinámicos en secuencias de video"

Con el siguiente contenido de capítulos:

I. Antecedentes.
II. Modelos basados en la corteza visual.
III. Método de segmentación de objetos dinámicos.
IV. Método de segmentación de objetos estáticos basado en enfoque híbrido.
V. Método perceptual neuroinspirado para segmentación de objetos estáticos.
VI. Discusión y conclusión.

Atentamente,

"La técnica por el engrandecimiento de México"

M.F. LUIS CARDONA CHACÓN
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SECRETARIA DE EDUCACION PUBLICA
INSTITUTO TECNOLOGICO DE CHIHUAHUA

LCCH/adcs.*.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Chihuahua

"2014, Año de Octavio Paz"

Chihuahua, Chih., 15 de septiembre de 2014

M.F. LUIS CARDONA CHACÓN
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

Por medio de la presente notificamos a usted que en cumplimiento parcial de los requerimientos para la obtención de grado de Doctorado en Ciencias en Ingeniería Electrónica, el documento de tesis del C. JUAN ALBERTO RAMÍREZ QUINTANA, ha sido aprobado y aceptado para su impresión. El título de la tesis es:

"Diseño de redes neuronales basados en modelos perceptuales multinivel para segmentación de objetos estáticos y dinámicos en secuencias de video"

Por lo que proponemos, le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

ATENTAMENTE
"LA TÉCNICA POR EL ENGRANDECIMIENTO DE MÉXICO"

DR. MARIO IGNACIO CHACÓN MURGUÍA
DIRECTOR DE TESIS

DR. MARCELINO ANGUIANO MORALES
MIEMBRO DEL JURADO DE EXAMEN

DR. JAVIER VEGA PINEDA
MIEMBRO DEL JURADO DE EXAMEN

DR. SERGIO D. CABRERA
MIEMBRO DEL JURADO DE EXAMEN

DRA. MARÍA DEL PILAR GÓMEZ GIL
MIEMBRO DEL JURADO DE EXAMEN

DRA. DIDIA PATRICIA SALAS PEIMBERT
MIEMBRO DEL JURADO DE EXAMEN

Chihuahua, Chih. A 20 de Septiembre de 2014

Mtro. Juan Manuel Cantú Vázquez.
Director general de DGEST.

At´n. Lic. Hanna de Lamadrid Téllez.
Coordinadora de la CNBES.

Presente

Por este conducto aprovecho la ocasión para saludarla e informarle que a la fecha he obtenido el Grado de Doctorado en Ciencias de Ingeniería Electrónica en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua. Motivo por el cual agradezco todo el apoyo brindado por esta Institución que Usted representa.

Agradezco a la Dirección General de Estudios Tecnológicos (DGEST) ahora Tecnológico Nacional de México, por el soporte institucional brindado.

Este trabajo fue resultado del proyecto Segmentación dinámica en videos mediante técnicas 2D y 3D que tiene como línea de investigación Procesamiento de señales eléctricas y electrónica y clave de registro CHI-IET-2012-105.

Atentamente

Juan Ramírez Q.

Juan Alberto Ramírez Quintana

A mi familia

El cerebro es una entidad muy diferente de las del resto del universo. Es una forma diferente de expresar todo. La actividad cerebral es una metáfora para todo lo demás. Somos básicamente máquinas de soñar que construyen modelos virtuales del mundo real.

Rodolfo Llinás Riascos

AGRADECIMIENTOS

A Dios, humildemente te doy gracias por darme la vida, la sabiduría, fortaleza y la oportunidad de realizar este proyecto.

A mis padres Ana y Juan, mis hermanas Ana Laura y Miriam Cecilia por ser mi principal apoyo e inspiración en todos los aspectos de mi vida. Agradezco a mis sobrinos Daira, Alex y mi sobrina-ahijada Ceci porque son parte muy importante de mi vida. A todos mis ti@s, prim@s que estuvieron pendientes de este proyecto realizado. Agradezco especialmente a mi hermana Miriam y mi tía Rocío por el apoyo extra recibido.

Al Dr. Mario Chacón por su enorme apoyo sin el cual no habría podido realizar este proyecto. Le agradezco también toda la sabiduría y conocimientos transmitidos así como sus valiosos consejos. Le gradezco también por su orientación, revisiones, sugerencias, así como el tiempo que invirtió en este proyecto.

A mi comité de tesis y jurado de titulación Dr. Mario Chacón, Dr. Javier Vega Pineda, Dr. Sergio D. Cabrera, Dra. Pilar Gómez Gil, Dr. Marcelino Anguiano Morales, Dr. Rafael Sandoval Rodriguez y la Dra. Didia Patricia Salas Peimbert por sus revisiones, sugerencias, así como el tiempo que invirtió cada uno en este proyecto.

Al personal de la DEPI y del ITCh que me apoyaron en la realización de este proyecto a lo largo de estos tres años y medio. Entre estos gracias a Dr. José Rivera, Maestra Blanca Lidia Lujan, Dr. Rafael Sandoval Rodríguez, Dr. Gaspar Jiménez, M.F. Luis Cardona, Alma Corral, Lucy Hernández, Nelly, Paco, Dr. Gerardo Trujillo, Ing. Floriano, Ing. Zambrano, F Aguirre, Carlos Méndez, Esperanza, Claudia Rojo, Yadira, Jorge Duran y Silvia Medrano.

A mis compañeras y amigas del doctorado Chelita, Paloma y Wendy. A mis amigos y compañeros de laboratorio Chelita, Jorge (Che), Raul, Kike, Huber, Llavero, Benja, Adrian, Sobe, Jorge Cardona, Omar Arias, Jorge Sagarnaga, David, Pepe y Leo. También gracias al

resto de todos los compañeros de la maestría que conocí a lo largo de estos tres años y medio, los cuales son muchos. Gracias a mis amigos de siempre Fredy, Miguel, Charly, Alan, Mauricio, Rocío Salcedo y Adriana.

A ti Elizabeth Anaya, te agradezco porque estuviste conmigo en casi toda esta etapa del doctorado. Te deseo que seas feliz siempre.

DISEÑO DE REDES NEURONALES BASADO EN MODELOS PERCEPTUALES MULTINIVEL PARA SEGMENTACIÓN DE OBJETOS ESTÁTICOS Y DINÁMICOS EN SECUENCIAS DE VIDEO

Juan Alberto Ramírez Quintana

División de Estudios de Posgrado e Investigación del

Instituto Tecnológico de Chihuahua

Chihuahua, Chih., 2014

Asesor: Ph. D. Mario Ignacio Chacón Murguía

El avance tecnológico en las áreas de la electrónica y la computación ha permitido el desarrollo de nuevos conocimientos en diferentes áreas como visión por computadora y neurociencias. Estas últimas, han generado una gran cantidad de información del funcionamiento del sistema nervioso y la corteza visual del cerebro, lo cual ha contribuido en el desarrollo de nuevos modelos y métodos basados en Redes Neuronales Artificiales (RNA) que son aplicadas a las áreas de visión por computadora, permitiendo el desarrollo de mecanismos similares a las funciones de la percepción visual.

En este trabajo de tesis, se propone un nuevo esquema para tareas de segmentación de objetos estáticos y dinámicos en secuencias de video. Sin embargo, en este caso se tomó como base se utilizó un conjunto de modelos y métodos, los cuales fueron diseñados en esta tesis a partir de un análisis del estado del arte de las RNA y modelos neurocomputacionales desarrollados a partir de las teorías más recientes del funcionamiento de la corteza visual del cerebro.

Las RNA propuestas son SOM Retinotópica (RESOM), Red Neuronal Celular de Binarización por Vecindario (NTCNN), la Red Neuronal Celular de Binarización por Vecinos Cuatro (T4CNN), Modelo Cortical Pulsante de la Gestalt (GSCM) y los filtros de color basados en Campos receptivos. A partir de estas RNA, se propusieron los métodos DR-SOM para modelado de fondo, SOM-CNN como segmentación de objetos dinámicos, KCNN utilizado en segmentación de objetos estáticos y PBSeg para segmentación de objetos estáticos y contornos. Todos los modelos y métodos fueron diseñados utilizando videos reportados en la literatura los cuales contienen escenarios reales.

Los resultados obtenidos por todos los métodos fueron exitosos. DR-SOM y SOM-CNN, se compararon con los métodos para detección de movimiento más recientes y mostraron tener un buen desempeño en escenarios con diferentes problemas, tales como fondos dinámicos y cambios de iluminación. Los métodos KCNN y PBSeg fueron comparados con otros métodos de segmentación de imágenes fijas y mostraron tener mejor desempeño para segmentar objetos con escenarios que varían en el tiempo. Adicionalmente, todos los métodos tuvieron tiempos de procesamiento que son compatibles con procesamiento de video en tiempo real, tanto en implementaciones en CPU como en CPU/GPU.

En consecuencia, los métodos de segmentación neuroinspirados desarrollados en esta tesis son factibles para distintas aplicaciones relacionadas a sistemas de vigilancia, robótica, medicina, etc.

NEURAL NETWORKS DESIGN FOR STATIC AND DYNAMIC OBJECT SEGMENTATION IN VIDEO SEQUENCES BASED ON MULTILEVEL PERCEPTUAL MODELS

Juan Alberto Ramírez Quintana

Chihuahua Institute of Technology

División de Estudios de Posgrado e Investigación

Chihuahua, Chih., 2014

Advisor: Ph. D. Mario Ignacio Chacón Murguía

The technological advances in electrical and computer areas have allowed new trends in neurosciences and computer vision. Neurosciences have developed a great amount of knowledge regarding the behavior of the nervous system and the cerebral cortex, which has contributed in the development of new models and methods based on Artificial Neural Networks (ANN) for computer vision applications. These ANNs have allowed the implementation of mechanisms similar to visual perception abilities.

This thesis, proposes a new scheme for static and dynamic object segmentation in video sequences. The basis of this work is a set of models and methods designed in this thesis from a state of the art analysis of ANN and neurocomputational models based on the visual cortex.

The ANNs proposed in this work are the Retinotopic SOM (RESOM), Neighborhood Threshold Cellular Neural Network (NTCNN), Threshold Cellular Neural Network by four-Neighborhood (T4CNN), Gestalt Spiking Cortical Model (GSCM) and the color filters based on color receptive fields. From these ANNs, a set of segmentation methods are proposed. These methods are, DR-SOM for background subtraction, SOM-CNN related to dynamic object segmentation, KCNN to achieve static object segmentation and PBSeg used for contour and static object segmentation. All the methods and models were designed considering popular videos reported in the literature which have real scenarios.

Results obtained by the methods DR-SOM and SOM-CNN were successful. These methods were compared with other state of the art methods for motion detection, and they showed very good performance in videos with dynamic background and illumination

changes. KCNN and PBSeg were compared with different methods designed for still image segmentation. According to the results, KCNN and PBSeg had the best performance in time-variant scenarios. Furthermore, all the methods achieved real-time processing in CPU and CPU/GPU implementations.

Based on the aforementioned facts, the neuroinspired segmentation methods developed in this thesis are feasible for applications related to surveillance systems, robotics, medicine, etc.

ÍNDICE

LISTA DE FIGURAS

I. ANTECEDENTES

II. MODELOS BASADOS EN LA CORTEZA VISUAL

III. MÉTODO DE SEGMENTACIÓN DE OBJETOS DINÁMICOS

IV. MÉTODOS DE SEGMENTACIÓN DE OBJETOS ESTÁTICOS BASADO EN ENFOQUE HÍBRIDO

V. MÉTODO PERCEPTUAL NEUROINSPIRADO PARA SEGMENTACIÓN DE OBJETOS ESTÁTICOS

VI. DISCUSIÓN Y CONCLUSIÓN

LISTA DE TABLAS

III. MÉTODO DE SEGMENTACIÓN DE OBJETOS DINÁMICOS

V. MÉTODO PERCEPTUAL NEUROINSPIRADO PARA SEGMENTACIÓN DE OBJETOS ESTÁTICOS

VI. DISCUSIÓN Y CONCLUSIÓN

I. ANTECEDENTES

Visión por computadora se refiere al procesamiento automático de imágenes digitales para extraer e interpretar información visual. Dicha área, se basa en el desarrollo de algoritmos que solucionan distintas tareas que implican la detección y reconocimiento coherente de objetos capturados en imágenes digitales. Para interpretar la información en imágenes digitales, los algoritmos tienen varias etapas que forman una cadena de procesamiento como la que se muestra en la Figura 1.1 la cual consiste de las siguientes etapas.

1) *Preprocesamiento.* Operaciones para adaptar la información de una imagen y tener mejor análisis en etapas posteriores del algoritmo. Ejemplos de preprocesamiento son las operaciones de brillo y contraste.

2) *Segmentación.* Operaciones para hacer una partición de la imagen en varias regiones que representen la información necesaria para el problema a resolver.

3) *Detección de objetos y clasificación.* Determinación y clasificación de los objetos contenidos en la imagen.

4) *Análisis de imágenes.* Obtener información de alto nivel acerca de lo que la imagen muestra.

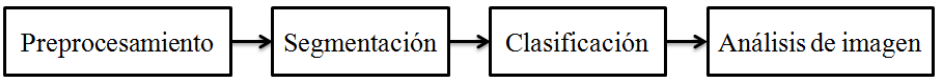


Figura 1.1. Etapas de procesamiento de imágenes digitales.

Este proceso, de interpretar imágenes, no es necesariamente lineal, ya que pueden existir elementos recurrentes dentro de cada etapa o entre las mismas etapas. Esta cadena de procesamiento se aplica en diferentes áreas como reconocimiento de patrones, reconocimiento de objetos y análisis de video, esta última es la que cobra mayor interés en este trabajo de tesis por las razones que se describen a continuación.

En visión por computadora, cada vez existe un mayor interés en utilizar distintas metodologías para cada una de las etapas del procesamiento digital de imágenes en aplicaciones como análisis de video; esto gracias al desarrollo de cámaras, sensores de adquisición de video, procesadores de bajo consumo de potencia y sistemas embebidos, los

cuales han permitido una mayor plausibilidad para realizar aplicaciones en tiempo real y con bajo costo [1]. Esto es un punto particularmente importante, ya que el análisis de video es una de las principales tareas de mayor interés en visión por computadora, pero anteriormente era complicado por las necesidades de tiempo cuando se procesa información entre un cuadro y otro. Sin embargo, con el surgimiento de tecnologías embebidas de bajo consumo de potencia como teléfonos inteligentes, tabletas, computadoras y otros dispositivos computacionales, actualmente es posible ejecutar algoritmos para análisis de video con carga computacional considerable [2]. Estas tecnologías se prestan para nuevos esfuerzos en aplicar modelos muy complejos en distintos tipos de hardware como en [3**,**4**,**5**]**. En consecuencia, la tecnología actual permite de forma simple implementar aplicaciones relacionadas al análisis de video, por lo tanto, la investigación relacionada a este tema debe ser en los próximos años un tema muy recurrente.

Uno de los aspectos que en ocasiones requiere carga computacional por los algoritmos y modelos que a veces se tiene que recurrir es la segmentación. Esta etapa del procesamiento de imágenes y video ha sido un tópico que en recientes años ha cobrado interés en visión por computadora, ya que es un paso fundamental para el entendimiento de un escenario (como se puede ver, en la Figura 1.1 es el segundo paso) [6**,**7**].** Al ser una etapa intermedia, la segmentación es un paso crítico en el procesamiento de imágenes y video, ya que permite realizar tareas complicadas en aplicaciones como seguridad, inspección de calidad en la industria, multimedia, análisis de patrones biométricos, etc. Si esta etapa inicial no genera buenos resultados, es imposible tener buenos resultados en las etapas finales [8]. Segmentación ha sido ampliamente estudiada en múltiples trabajos en la literatura, aunque el uso de Redes Neuronales Artificiales (RNA) para tareas de segmentación ha sido uno de los tópicos estudiados para solucionar tareas de segmentación de video e imágenes [9]. De acuerdo al análisis de literatura en [9], uno de los enfoques más recientes es el de desarrollar métodos de segmentación basados en modelos bioinspirados y teorías de percepción. Además, debido al inherente paralelismo del procesamiento neural, las RNA son modelos que pueden ser utilizadas para resolver problemas de segmentación y procesamiento de video muy complejos y obtener resultados para aplicaciones en tiempo real en distintos tipos de hardware [5].

Por lo tanto, dado que la etapa de segmentación es uno de los pasos más críticos en la cadena de procesamiento para interpretar la información en una imagen, y aunado a esto, el análisis de video es un tema que ha sido particularmente importante en visión por computadora debido al amplio desarrollo tecnológico actual, en este trabajo de tesis el enfoque es la segmentación en secuencias de video. El resultado de dicha segmentación es partir un escenario en un conjunto de regiones que representen los objetos contenidos en él. No obstante, el reto en la segmentación en una secuencia de video, es que los objetos pueden comportarse de diferentes maneras, ya que pueden estar en movimiento, ser fijos, pueden tener una trayectoria en particular, ser objetos de no interés, etc.

En la literatura, el trabajo relacionado a segmentación en secuencias de video se enfoca principalmente en aplicaciones relacionadas a detección de movimiento, aunque existen algunos trabajos relacionados con atención visual y segmentación de color. Respecto a los trabajos relacionados a detección de movimiento, están aquellos enfocados a detectar objetos dinámicos en sistemas de vigilancia, donde un escenario con cámara estacionaria adquiere video para detectar personas, carros u objetos sospechosos. Los trabajos relacionados con sistemas de vigilancia generalmente realizan la segmentación mediante modelos probabilísticos como en [10,11,12] donde se recurre a Modelos de Mezclas Gaussianas (GMM), filtros *Kalman* o Redes Bayesianas. En esta misma aplicación, destaca también el uso de RNA como en [7,13,14,15,16] donde redes como los Mapas Auto-organizados de *Kohonen* (SOM), Redes Neuronales Probabilísticas (PNN) y Redes Neuronales Celulares (CNN) se utilizan en el proceso de segmentación de objetos dinámicos. De acuerdo a la literatura analizada en esta tesis, los métodos más comunes para segmentación en sistemas de vigilancia son los basados en RNA y los métodos probabilísticos. Sin embargo, de acuerdo a [8], también existen muchos otros métodos para segmentación de objetos en sistemas de vigilancia como agrupamiento, flujo óptico, etc.

Se pueden encontrar en la literatura otras aplicaciones relacionadas a segmentación de video como análisis de movimiento en superficies de profundidad como en [17], atención visual de objetos sobresalientes en la escena como en [18], segmentación de color en secuencias de video para reconocer los objetos en el escenario como en [19,20].

Aunque existen muchos métodos y modelos para segmentación de objetos en secuencias de videos como se puede evidenciar en [8], y de acuerdo a **[**9**,**21] la segmentación basada en RNA es y ha sido un esquema muy exitoso ya que tal como se mencionó anteriormente, permiten resolver problemas muy complejos y en ocasiones con mejores tiempos de procesamiento que otros esquemas. Por lo tanto, en este trabajo de tesis la segmentación de objetos en secuencias de video se realizará basándose en RNA. En los análisis del estado del arte planteados en **[**9**,**22] realizados entre otros por el autor de esta tesis, se pudo observar que la tendencia en las RNA y visión por computadora es acercarse más a los modelos neuroinspirados derivados de áreas como neurociencias y psicología. Sin embargo, en el caso de los trabajos relacionados a segmentación en video, el enfoque se limita al análisis de la información del escenario sin tomar en cuenta muchos de los principios derivados de teorías neuroinspiradas. No obstante, después de hacer un análisis de literatura relacionado a modelos y teorías de percepción neuroinspirada, se planteó una hipótesis inicial en este trabajo de tesis, la cual es:

1). Los modelos que explican fenómenos de percepción visual derivados de las neurociencias y psicología que han surgido recientemente pueden ser útiles también para resolver problemas de percepción en secuencias de video.

Esta hipótesis surge a partir del estado del arte realizado en esta tesis sobre modelos neurocomputacionales que explican fenómenos perceptuales al nivel de la corteza visual, los cuales se abordarán posteriormente. También se generó una segunda hipótesis la cual es la siguiente:

2). En el análisis de video, la información de los objetos que están fijos puede llegar a ser tan útil como la información de los objetos dinámicos.

En los métodos clásicos de segmentación en secuencias de video la información analizada son los objetos dinámicos, descartando la información del fondo. Sin embargo, esta segunda hipótesis surge a partir de diferentes observaciones; por ejemplo, en el caso de sistemas de vigilancia, si en un video se muestra un escenario en el que una persona se detiene y deja una mochila sospechosa sobre el piso, esta se convertirá en un objeto que se vuelve fijo y no importa que tan robusto sea la segmentación de objetos dinámicos, la mochila se perderá después de un intervalo de tiempo. Por otro lado, si la información de

fondo también es segmentada, entonces se detectará que se agregó un objeto nuevo en los objetos estáticos, de manera que esta información no se pierde. Otro ejemplo puede ser en robótica, donde la segmentación de objetos estáticos puede servir para que un robot móvil analice un escenario para detectar si encuentra lo que busca. En general, segmentar los objetos fijos puede ser útil para analizar alguna característica del escenario, por ejemplo un objeto en particular, revisar si la cámara está mal ubicada, analizar cambios en el escenario al trascurrir el tiempo, etc. Por tanto, la información de los objetos de fondo puede ser útil en la interpretación de un escenario, dándole más utilidad a cualquier aplicación relacionada con análisis de video, y bajo este argumento, se plantea en esta tesis la necesidad de realizar investigación en segmentación de secuencias de video no solo para analizar objetos en movimiento, también se plantea desarrollar nuevos esquemas de segmentación en secuencias de video donde se analicen objetos fijos y en movimiento.

En consecuencia, basándose en las hipótesis planteadas, en esta tesis se propone el diseño de un conjunto de métodos de segmentación de objetos estáticos y dinámicos en secuencias de video. Estos métodos están basados en las teorías más recientes relacionadas al funcionamiento de la corteza visual y los modelos perceptuales. Para basarse en estas teorías, se analizan los modelos neurocomputacionales y neurocognitivos que se utilizan para entender los mecanismos perceptuales derivados de la corteza visual. El resultado de dicho análisis se utilizará para proponer un conjunto de RNA y modelos neuroinspirados, los cuales a su vez son la base para los métodos de segmentación.

Para desarrollar este trabajo de tesis, se tiene que analizar el actual estado del arte de los modelos neurocomputacionales relacionados al funcionamiento de la corteza visual y neurocognitivos, así como las teorías más recientes relacionadas a percepción.

1.1 Percepción Visual, Neurociencia Computacional

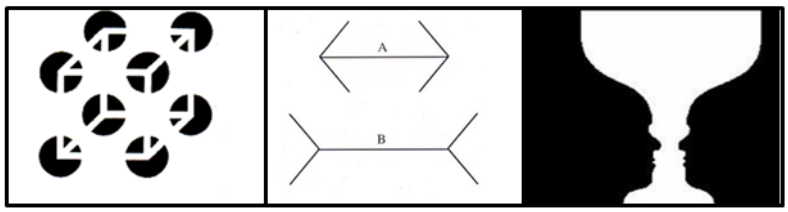
Psicología y neurociencias son áreas que continuamente han sido utilizadas para el desarrollo de nuevos esquemas, teorías y modelos en visión por computadora. En consecuencia, estas áreas han servido para generar algoritmos y modelos que imitan los procesos de visión biológica que existen en humanos y animales. Los avances en los últimos años en las áreas de neurociencias, han permitido tener diversas teorías acerca del funcionamiento de los mecanismos de percepción sensorial a nivel del sistema nervioso.

Dichos avances, han sido útiles en el desarrollo de modelos que brindan una explicación al procesamiento que realiza el sistema nervioso para interpretar la información que viene de cada uno de los sentidos; como en el caso de la visión. Sin embargo, también se han aprovechado para generar modelos y algoritmos bioinspirados para distintas áreas como visión por computadora y procesamiento digital de imágenes, cuya tendencia es imitar los mecanismos de percepción visual. Para entender estas áreas, primero es necesario conocer los conceptos de Percepción Visual y Neurociencia Computacional.

1.1.1 Percepción Visual

Percepción Visual es la habilidad para interpretar la información que proviene de los ojos. Dicha habilidad, brinda facilidad para tener conocimiento de lo que existe a nuestro alrededor con la luz que se refleja en las superficies. Esta forma de interpretación consiste en dividir una escena en cada uno de sus elementos visuales formando objetos coherentes. Estos elementos visuales se pueden detectar a partir de la visión del color, espacio, visión temporal, percepción de movimiento y de profundidad **[**23]. Esta habilidad se forma en un proceso que inicia en la retina y luego pasa a una parte del cerebro que se le conoce como corteza visual [24]. Mediante el proceso que se da en esta área del cerebro, se puede asimilar la información visual que viene del exterior. Sin embargo, para lograr la percepción, es necesario hacer en tiempo real tareas complicadas tales como detección de líneas, formas, movimiento, profundidad, texturas y colores.

Un aspecto interesante de la percepción visual es que no solamente nos permite reconocer nuestro entorno, también permite reconocer patrones de formas que no necesariamente se completan a partir de la información visual, como se ve en la Figura 1.2a donde se puede apreciar un cubo encima de círculos; aunque éste no se formó con base a los patrones físicos de luz, o en la Figura 1.2b se observan líneas de la misma longitud pero parecen de distinto tamaño, o en 1.2c se aprecian dos figuras en una sola imagen. Estas ilusiones son utilizadas para analizar la manera en que el proceso de percepción visual forma estímulos sensoriales que no existen [25].



a) b) c)

Figura 1.2. Fenómenos de la percepción[1]. a). Cubo que no existe. b). Ilusión Müller-Lyer; ilusión de las líneas c). Multiestabilidad en la imagen; dos figuras percibidas.

Existen varias teorías en las áreas de neurociencias y psicología para explicar las diversas habilidades de la percepción. Una de las más antiguas y aceptadas es la teoría de la *Gestalt*. Dicha teoría ha servido de inspiración y referencia para desarrollar modelos y mecanismos que explican la Percepción Visual, ya que estudia cómo las personas perciben los distintos componentes visuales [26]. El término *Gestalt* se puede traducir como "esencia o forma de una entidad" [27]. El principio de esta teoría indica que el cerebro procesa la información de manera holística, paralela y puede aprender por sí mismo. Las cinco principales leyes o principios en que procesa el cerebro la información visual se describen a continuación:

1) Proximidad. Este principio permite agrupar elementos de acuerdo a su cercanía.

2) Similaridad. Si las distancias entre los elementos son las mismas, entonces, nuestra mente agrupa elementos con la misma forma en una sola entidad.

3) Continuidad. Elementos que aparecen seguidos en una sola dirección son agrupados en un solo patrón.

4) Simetría. Imágenes que tienen formas muy similares son percibidas como iguales.

5) Pregnancia. Todo patrón tiende a percibirse con la forma resultante más relevante de todas.

La Figura 1.3, muestra el efecto de los principios de la *Gestalt*. Se puede apreciar en la Figura 1.3a tres barras formadas a partir de varias líneas, en 1.3b se observan dos triángulos a partir de los rombos y los círculos, en la Figura 1.3c, se forma un triángulo a partir de la continuidad de las líneas, en la Figura 1.3d dos figuras encontradas ligeramente distintas aparentan ser iguales y en la Figura 1.3e un círculo grande sobresale primero que los demás.

[1]Las ilustraciones de la Figura 1.2 están disponibles en diferentes sitios web.

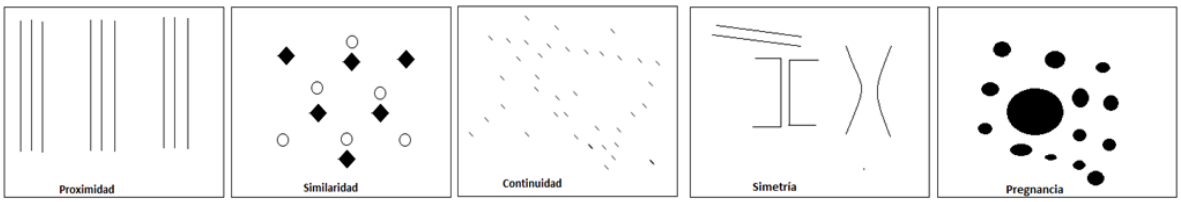


Figura 1.3. Ilustración de los principios de la Gestalt. a). Proximidad. b). Similaridad. c). Continuidad. d). Simetría. e). Pregnancia.

1.1.2 Neurociencia Computacional

Neurociencia computacional se refiere al estudio de la estructura y las funciones del sistema nervioso central y periférico en términos de las propiedades del procesamiento de la información [28]. Esta parte de las Neurociencias es en realidad un área interdisciplinaria que envuelve diferentes campos como anatomía, fisiología, psicología, ingeniería eléctrica, ciencias computacionales, modelado matemático y físico. La neurociencia computacional tiene varias líneas [29] de investigación destacando las siguientes:

Modelado de neurona. Se refiere al análisis de las características biofísicas y eléctricas de una neurona para obtener modelados de la estructura y el comportamiento de cada neurona como unidad dentro de una red que representa una capa cortical[2]. Entre los primeros ejemplos de modelado de neurona simple está el modelo de *Hodgkin-Huxley*, el cual consiste en simular la reacción eléctrica de un potencial de membrana [30].

Procesamiento sensorial. Esta línea estudia la manera en cómo el sistema nervioso responde ante estímulos sensoriales. En esta línea se estudian diversos fenómenos que generan las habilidades de percepción.

Memoria, plasticidad sináptica y aprendizaje. Estudia los mecanismos que generan los procesos de aprendizaje. Un aspecto interesante es el dilema plasticidad-estabilidad, el cual consiste en analizar la manera en como un sistema receptivo pueda aprender nuevos patrones, pero manteniendo la estabilidad cuando se presentan patrones irrelevantes. Uno de los primeros modelos de aprendizajes es la regla Hebbiana propuesta por Donald Hebb en 1949 [31].

Modelos de Redes. Se refiere al estudio de las interconexiones entre neuronas, capas corticales y circuitos neuronales. El modelado de redes permite explorar el potencial

[2] Cortical significa relativo a la corteza cerebral.

computacional tales como conectividad, simulación y análisis [31]. A partir de este análisis se generan múltiples modelos que han servido para simulación de mecanismos corticales, conocer la estructura de partes del sistema nervioso, mecanismos de aprendizaje, etc. Aunado a esto, una aplicación interesante del modelado de redes es el desarrollo de redes de neuronas artificiales para procesamiento de información, las cuales son las Redes Neuronales Artificiales (RNA). Estas últimas tienen un enfoque muy distinto en su aplicación a los modelos para conocer el comportamiento biológico del sistema nervioso, sin embargo, parten de los mismos paradigmas del modelado de redes neuronales biológicas. Actualmente, la neurociencia computacional ha impactado en el desarrollo de modelos que en los últimos años han permitido formar teorías acerca de la manera en cómo funcionan los fenómenos perceptuales, habilidades cognitivas y aprendizaje. Una de las áreas más estudiadas por modelos neurocomputacionales es la corteza visual y a partir de estos modelos se han generado múltiples teorías no solo respecto al funcionamiento de la corteza visual, también han servido para entender las interconexiones en las distintas capas corticales en el cerebro, así como el funcionamiento de capas corticales sensoriales. En el siguiente capítulo se documenta un análisis del estado del arte de los modelos neurocomputacionales basados en áreas corticales de la corteza visual. A partir de muchos de estos modelos, han surgido RNA que han sido exitosamente aplicadas en áreas como procesamiento de imágenes, visión por computadora, reconocimiento de patrones, etc.

Para esta tesis, los modelos neurocomputacionales basados en la corteza visual tienen gran importancia, ya que a partir de estos se pueden modelar RNA que puedan ser aplicados a tareas de segmentación en secuencias de video. La estructura de la tesis es la siguiente: en el capítulo II se documenta un estado del arte de los modelos Neurocomputacionales y RNA basados en la corteza visual. A partir de algunos modelos documentados en el capítulo II, en el capítulo III se presenta el modelado de dos RNA propuestas en estas tesis, además, en este mismo capítulo se presenta el método de segmentación de objetos dinámicos. En el capítulo IV se presenta un método de segmentación de objetos estáticos basado en RNA y agrupamiento. En el capítulo V se presenta un método de segmentación de objetos estáticos basado en modelos perceptuales, neurocognitivos y neurocomputacionales. Entre estos modelos documentados en el capítulo V, se propone un modelo de RNA para filtrado de color basado en el modelado de campos

receptivos y también se propone una RNA basada en principios perceptuales para obtener las regiones que representan los objetos.

II. MODELOS BASADOS EN LA CORTEZA VISUAL

En este capítulo se presenta un compendio de los modelos neurocomputacionales y RNA basados en el funcionamiento y la estructura de la corteza visual. Algunos de estos son utilizados en esta tesis como antecedente para desarrollar otros modelos de RNA factibles para análisis de video y métodos de segmentación de objetos estáticos y dinámicos en secuencias de video. En la sección 2.1 se presenta una introducción a la estructura de la corteza visual. En la sección 2.2 se presentan los modelos neurocomputacionales basados en la Teoría de Resonancia Adaptiva. En la sección 2.3 se presenta las RNA basadas en modelos pulsantes. En la sección 2.4 se presentan los modelos neurocomputacionales basados en Auto-organización. En la sección 2.5 se presenta el análisis de modelos neurocognitivos. Finalmente en la sección 2.6 se presentan las conclusiones de los modelos analizados en este capítulo.

2.1. Corteza Visual

Las habilidades de la percepción visual se generan principalmente en un área del cerebro que se le conoce como corteza visual. Esta área del cerebro se compone de la retina, los núcleos laterales geniculados (LGN) y varias áreas de procesamiento visual las cuales son la corteza visual primaria (V1), el área visual 2 (V2), área visual 4 (V4), la corteza medio temporal (MT), Parietal (ParC), temporal (TempC) y prefrontal (PrFC) [32]. En la Figura 2.1 se puede observar un diagrama general que muestra los principales elementos de la corteza visual y la manera en que se interconectan para desarrollar la percepción de objetos, a continuación se describen las principales partes.

2.1.1 Retina

Tejido neuronal que se encuentra en el ojo y convierte la luz en señales nerviosas. Consta de varias capas, de las cuales, las más importantes son la capa de fotoreceptores (conos y bastones), células bipolares y células ganglionares [33]. Dentro de los fotoreceptores los bastones proveen una representación de la luz, mientras que los conos proveen una representación de los colores. Existen tres tipos de conos separados en *Long* (L), *Medium* (M) y *Short* (S), basándose en sus funciones de sensibilidad. Es decir, los conos *L* son sensibles a las longitudes de onda largas, por lo que son selectivos a los rojos,

los conos *M* son selectivos a los tonos verdes y los conos *S* son selectivos a los azules. El comportamiento de la sensibilidad de los conos se puede ilustrar en la Figura 2.2. En la salida de la retina, se encuentran las células gangliares (RGC), las cuales tienen células que responden a polaridades de luz. Estas se pueden clasificar en dos tipos:

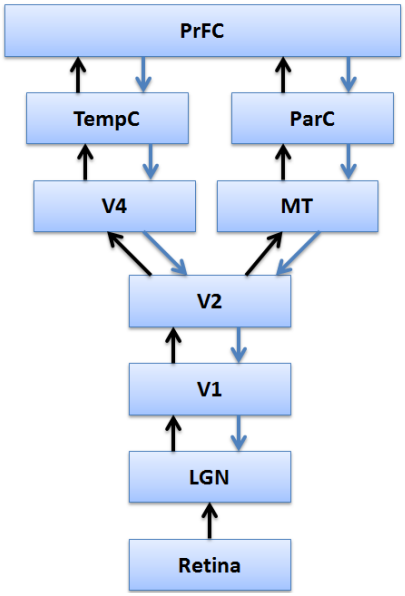


Figura 2.1. Parte de la corteza visual dedicada al procesamiento de atención visual y percepción [32].

Centro ON: Responden a polaridades donde se tiene luz rodeada de obscuridad, como en la Figura 2.3a.

Centro OFF: Responden a polaridades donde se tiene obscuridad rodeada de luz, como en la Figura 2.3b.

Las RGC tienen un conjunto de axones que forman el nervio óptico y se extienden hasta el tálamo.

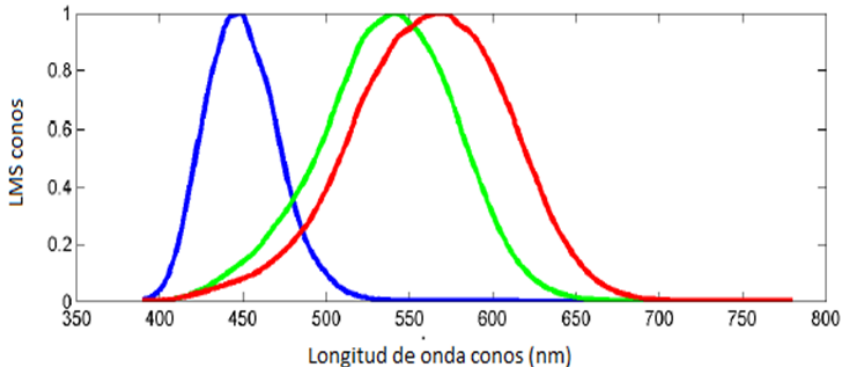


Figura 2.2. Funciones de longitud de onda para L (rojo), M (verde) y S (azules).

2.1.2 Nervio óptico y los Núcleos Laterales Geniculados (LGN).

Las fibras del nervio óptico llegan al cerebro y pasan sin interrupción al quiasma óptico[3]. Allí, aproximadamente la mitad de las fibras cruzan al lado opuesto del ojo de donde proceden, y la otra mitad permanecen en el mismo lado, como se muestra en la Figura 2.4. Desde el quiasma las fibras continúan a diferentes partes del tálamo[4] conocidas como Núcleos Laterales Geniculados. Estos últimos son células aferentes que envían las señales del nervio óptico a la corteza cerebral, y también reciben fibras de otras áreas del cerebro de la corteza cerebral para retroalimentar procesos de atención y activación.

Figura 2.3. Respuesta de las RGC en la retina [34]. a). Células ON. b). Células OFF.

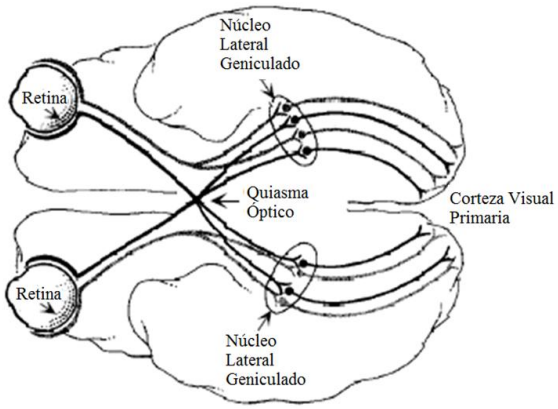


Figura 2.4. Camino de la retina hasta la corteza visual pasando por los LGN [31].

Las neuronas en los LGN tienen propiedades similares a las RGC y tienen una estructura retinotópica. En múltiples estudios se ha visto que los campos receptivos (RF) de RGC/LGN pueden ser modelados por gaussianas con distintas varianzas. Existen dos tipos de LGN los Centro/Alrededores (C/A) y los oponentes (Ce). Los C/A son RF que se pueden modelar con la diferencia de gaussianas, y su función es filtrar la información de forma tal que solo se estimule la diferencia en polaridades de luz. Estos tienen una

[3]Quiasma óptico es la parte donde se cruzan las fibras del nervio óptico.
[4]El tálamo es una parte del cerebro por donde pasan todos los estímulos sensoriales excepto el del olfato. Su función es recibir información para conducirla a la corteza cerebral y filtrar la información "trivial".

distribución como los RGC por lo que existen los LGN ON y LGN OFF como los de la Figura 2.3. Los oponentes, tiene neuronas espectralmente oponentes, de forma tal que funcionan como filtros que pasan ciertas longitudes de onda **[**35**,**36].

De los principales elementos en las primeras etapas de percepción en el cerebro, están los campos receptivos (RFs), los cuales son neuronas que alteran su respuesta en función de un estímulo sensorial. Estas células se encuentran en áreas como la retina, los LGN y la corteza visual. En el caso de las neuronas visuales, un campo receptivo es una porción de la retina que cuando es estimulado causa un cambio en el disparo de una neurona dada.

Las células en los LGN y otras partes de la corteza se dividen en dos clases: Células simples y Células complejas. Las Células simples generan una respuesta basada en combinaciones lineales que se pueden describir mediante operaciones como la convolución, mientras que las Células complejas tienen respuestas que se describen mediante operaciones no lineales [31].

2.1.3 Corteza Visual

Es la parte de la corteza cerebral dedicada a procesar la información visual. Se localiza en el lóbulo occipital del cerebro, y se divide por áreas visuales. La corteza visual primaria (V1) es el área más estudiada de la corteza visual, localizada en la parte posterior del cerebro y se divide en 2500 módulos que preprocesan la información espacial en el campo visual [37]. En V1 existen RF que también tienen una estructura retinotópica (generan una representación de la imagen formada en la retina) y son selectivos a ciertas propiedades como orientación, frecuencia espacial, dominancia ocular y color [34]. En la Figura 2.5a y 2.5b se muestran ejemplos de células selectivas a orientación, mientras que en 2.5c y 2.5d se muestran células espacio-temporales.

Figura 2.5. Campos receptivos en V1. a). RF selectivo a 45º. b). RF selectivo a 135º. c). RF en el tiempo 0. d). RF de c) en el tiempo 1.

El área visual 2 es la segunda área mejor conocida de la corteza visual. Esta área tiene conexiones directas con V1 y envía conexiones a otras áreas visuales. Se divide en

cuadrantes que sirven para proveer un mapeo de la información visual tal como en V1, pero tiene cierta modulación en el proceso de atención. Otras áreas visuales son V4, relacionada con la detección de objetos, y el área MT, asociada al procesamiento de movimiento. La corteza visual se auxilia de otras áreas tal como se observa en la Figura 2.1 donde se ve que las áreas de la corteza visual se relacionan con otras áreas como la corteza prefrontal (PrFC), parietal (ParC) y temporal (TempC). Esto con el fin de retroalimentarse en el proceso de atención visual. La corteza prefrontal y parietal se encarga de hacer el procesamiento de atención y decisión, los cuales retroalimentan a ciertas áreas de la corteza visual incluyendo a los LGN.

2.1.4 Modelos Neurocomputacionales

En la literatura se reporta una gran cantidad de modelos neurocomputacionales basados en las áreas visuales del cerebro. La mayor parte se enfoca a analizar el funcionamiento de las partes más bajas de la corteza visual. No obstante, estos modelos tratan a su vez de explicar la manera en como el cerebro procesa la información para generar las habilidades de percepción. Para entender el estado actual de estos modelos, se realizó un análisis de literatura que comprendió la búsqueda de investigadores de temas relacionados, laboratorios de investigación, reportes técnicos y artículos técnico-científicos sobre visión por computadora, redes neuronales artificiales y percepción visual. En este análisis se encontraron diferentes modelos que se pueden clasificar de la siguiente forma:

- Teoría de Resonancia Adaptiva.
- Redes Auto-organizadas.
- Redes Neuronales Pulsantes.
- Esquemas Neurocognitivos.

El primer caso, se refiere a todas las RNA y modelos neurocomputacionales que se desarrollaron a partir del concepto de resonancia adaptiva [38]. Las redes auto-organizadas incluyen a los Mapas Auto-organiados de *Kohonen* y el modelo Mapa Auto-organizado Sinérgicamente y Lateralmente Conectado (LISSOM). Las Redes pulsantes, agrupan todas aquellas neuronas cuya salida es una señal en el dominio del tiempo. Los esquemas neurocognitivos son aquellos que modelan el comportamiento del pensamiento y la percepción con capas corticales representadas con cierto nivel de abstracción [39]. A

continuación se describen los trabajos más relevantes derivados de modelos neurocomputacionales y RNA que se inspiran a partir del comportamiento de partes corticales de la corteza visual. Estos trabajos sirvieron posteriormente para el desarrollo de nuevos modelos que serán aplicados en los distintos métodos de segmentación propuestos.

Como nota aclarativa los modelos que no fueron implementados (solo fueron analizados) serán presentados en esta tesis conservando la notación con la que se presentaron en sus respectivas referencias. Para los modelos que si fueron implementados en forma total o parcial, en esta tesis se generó una notación nueva.

2.2. Teoría de Resonancia Adaptiva

La Teoría de Resonancia Adaptiva (ART) fue desarrollada por Stephen Grossberg y Gail Carpenter [38]. ART hace referencia a un conjunto de RNAs supervisadas y no supervisadas que resuelven el dilema plasticidad-estabilidad que existe en el proceso de aprendizaje de las RNA. El dilema plasticidad-estabilidad trata de que un sistema receptivo pueda aprender nuevos patrones, pero manteniendo la estabilidad cuando se presentan patrones irrelevantes [38]. Aún cuando en la literatura es común encontrar diversas aplicaciones con redes basadas en ART como se puede ver en [9], esta teoría no se limita únicamente a RNA; en realidad se refiere a la relación que existe en las diversas capas del cerebro desde el tálamo hasta las áreas relacionadas con la memoria. Un enfoque planteado por Stephen Grossberg es la resonancia adaptiva en las capas de la corteza visual, el cual implica definir el funcionamiento de los distintos niveles de la corteza visual que van desde los LGN hasta la corteza prefrontal en términos de un conjunto de capas corticales jerárquicamente relacionadas. Para ello, en el Laboratorio de Percepción Cognitiva de la Universidad de Boston han desarrollado diversos modelos neurocomputacionales que van explicando la formación de objetos, movimiento y almacenamiento de información en la corteza visual. En este trabajo se analizarán los modelos basados en análisis de objetos y de movimiento, como LAMINART y FORMOTION.

2.2.1 LAMINART

LAMINART es un modelo que explica cómo se lleva a cabo la detección de objetos mediante los mecanismos de agrupamiento perceptual, detección de profundidad y atención en la corteza visual. Inicialmente, LAMINART se desarrolló para describir el

procesamiento de objetos [40], pero después se propuso LAMINART 3D en el que se explican las representaciones de bordes, formas y superficies en áreas de niveles corticales LGN, V1, V2 y V4 con imágenes 2D de superficies en 3D. De manera que LAMINART 3D [41] expone la forma en cómo las características monoculares adquiridas en la retina se van fusionando para generar una representación binocular de superficies en espacios tridimensionales [40].

Un diagrama del modelo LAMINART [41], se muestra en la Figura 2.6, donde el nivel V1 tiene dos etapas corticales denominadas 4 y 3B con un conjunto de células simples, las cuales son células que responden a bordes y líneas con una orientación particular. Además, en el diagrama de LAMINART se muestra una etapa 2/3A que contiene células complejas que también son sensibles a los bordes pero con cierto grado de invariancia espacial, es decir, filtran bordes irrelevantes según la respuesta de los campos receptivos. En el nivel V2, se observa un procesamiento dado por células complejas en cada etapa, además de que existe un filtro de disparidad que ayuda al procesamiento de profundidad. En el nivel V4, es donde se completa el procesamiento que resulta en la segmentación de superficies en espacios de profundidad.

También se puede observar en la Figura 2.6 que el modelo LAMINART presenta una retroalimentación constante entre los niveles V1 y V2; esto es debido a que la hipótesis que motivó el desarrollo del modelo, menciona que mediante la percepción de bordes se va haciendo la estereopsis[5] [41]. El modelo LAMINART tiene cinco redes que son distribuidas en los niveles V1, V2 y V4 de la corteza visual, los cuales son V1 bordes monoculares, V2 bordes binoculares, V2 bordes, V2 superficies monoculares y superficies V4. A continuación se abordará una breve descripción del modelo LAMINART, el modelo completo se puede analizar en [41], y la notación para LAMINART conservan las misma notación que en [41] en esta tesis.

V1 bordes monoculares. Primero la entrada es procesada por la retina y los LGN, los cuales dejan pasar la información del centro del vecindario de cada receptor (pixel) que viene de la retina, e inhibe la información de los alrededores de dicho vecindario. En este modelo, los LGN se representan por:

[5]Estereopsis es el fenómeno por el cual se puede formar la percepción de profundidad a partir de dos imágenes.

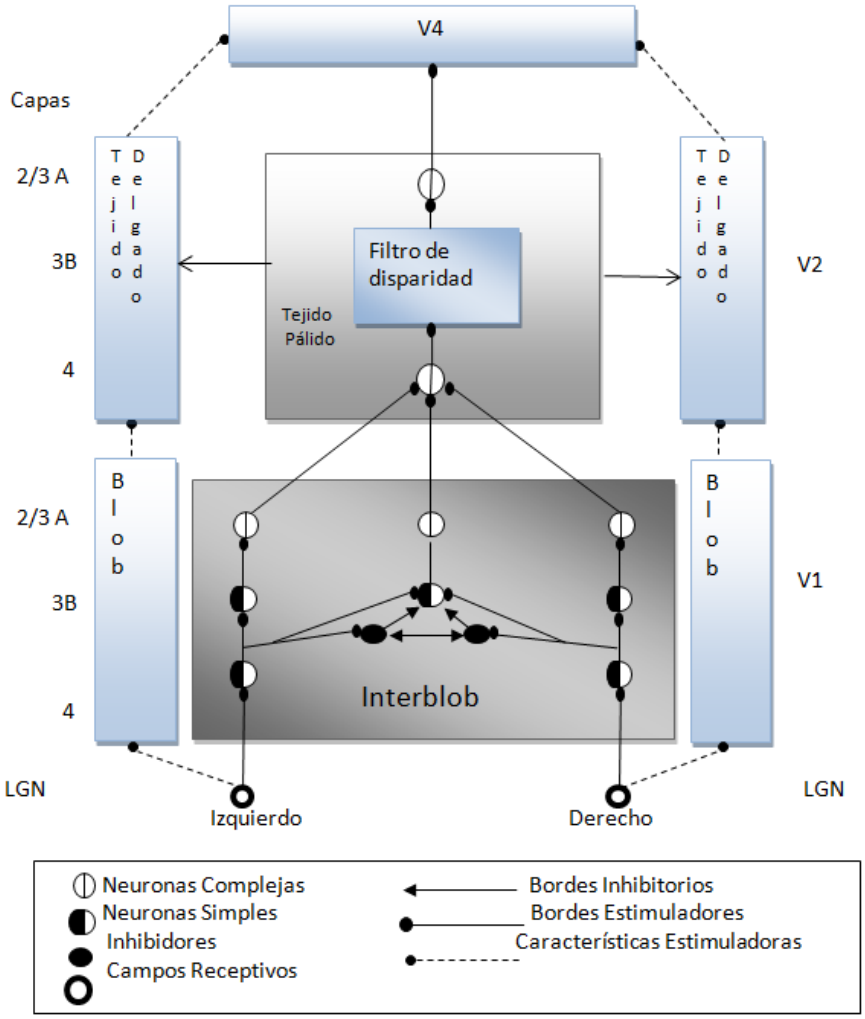


Figura 2.6. Diagrama del modelo LAMINART **[**41**]**.

$$\frac{dx_{ij}^{L/R}}{dt} = -\alpha x_{ij}^{L/R} + \left(\beta - x_{ij}^{L/R}\right) I_{ij}^{L/R} - x_{ij}^{L/R} \sum_{p\neq i, q\neq j} G_{pqij} I_{pq}^{L/R} \tag{2.1}$$

donde *L/R* designa que la célula pertenece a la retina izquierda (*L*) o derecha (*R*), *(i,j)* la coordenada del píxel, (*p,q*) índice de la célula, α,β son constantes del potencial de membrana, $x_{ij}^{L/R}$ el estado de los LGN, $I_{pq}^{L/R}$ es la luminancia en la retina, G_{pqij} es un kernel gaussiano que hace la inhibición en las partes alrededor del centro. Luego, el procesamiento de los bordes monoculares se desarrolla en el área de 4, 3B y 2/3 del área interblob de V1 (Figura 2.6). En la capa 4 se reciben las entradas y las células simples actúan como filtros de orientación sensitivos a los cambios de polaridades de contraste. El procesamiento de la capa 4 está dado por:

$$s_{ijk}^{L/R+} = \sum_{p,q} K_{pqk} \left[x_{i+p,j+q}^{L/R} \right] \tag{2.2}$$

donde k denota orientación y K_{pqk} es una función de Gabor que representa cada campo receptivo. Esta función es para los cambios de polaridad de obscuro a luz. Si se tiene la polaridad de contraste contraria, el resultado se cambia de signo y a ésta se le denomina $s_{ijk}^{L/R-}$. La capa 3B solo hace un procesamiento dado por:

$$b_{ijk}^{L/R+/-} = 2\left[s_{ijk}^{L/R+/-} \right] \tag{2.3}$$

La capa 2/3 tiene un conjunto de células complejas que responden a todos los bordes de objetos si tienen un contraste polarizado de manera inversa con respecto al fondo, haciendo la implementación de detección de bordes invariantes al contraste. Esta capa implementa un temprano agrupamiento perceptual; es decir, se forma un proceso de seguimiento de bordes permitiendo formar los contornos de los objetos detectados. La dinámica de las células 2/3 se describe en V1 bordes binoculares.

***V1 bordes binoculares*.** Se localiza en el área interblob e incluye las células binoculares de las capas 3B, 2/3. La capa 3B inicia el proceso de fusión de bordes para estereopsis. Las células de V1 asociadas a la retina izquierda y derecha en la capa V1, 3B activan las células binoculares en la capa 3B cuya respuesta es retroalimentada por la salida en la capa 4. La capa 3B también contiene un conjunto de neuronas que hacen una inhibición si las entradas difieren mucho en magnitud entre sí. Además, la capa 3B es sensitiva a la misma posición y disparidad pero opuesta al conjunto de polaridades de contraste de las células 2/3 binoculares. Como en el caso monocular, la capa 2/3 implementa una detección de bordes invariantes al contraste y hace un agrupamiento perceptual, pero éste es selectivo a la disparidad. El modelo de esta capa se puede analizar en [41].

***V2 Bordes*.** Esta parte se encuentra en la región de V2 e incluye células binoculares en las capas 4 y 2/3 de V2. Las entradas monoculares y binoculares son combinadas en la capa 4 de V2 y las células monoculares no están asociadas con un plano de profundidad en particular. La retroalimentación superficie-bordes es modulada en la capa 4 de V2 y va estimulando la actividad de dicha capa al recibirse las señales de superficie-bordes en ambos lados o se suprime de otra manera. En la capa 3B de V1 se relacionan los bordes verticales en una imagen de la retina con los bordes de la otra retina. Como resultado, se

pueden generar falsos bordes, para lo cual en V2 existe un filtro de disparidad simétrico sobre un plano fijo para inhibir aquello que esté lejos de dicho plano. En la Figura 2.7 se observa un ejemplo del funcionamiento de dicho filtro, donde se ve que los puntos formados en el plano fijo corresponden a los bordes reales, mientras que los bordes formados en el resto de los planos tienen puntos que representan bordes falsos. El filtro ayuda con el problema de correspondencia y se asume como un control inhibitorio de bordes falsos. En V2 Bordes va resultando un conjunto de bordes en el que se van conectando sus puntos para formar los objetos bajo un agrupamiento similar a los principios de la teoría de la *Gestalt*. En [41] se puede analizar el modelo de esta área.

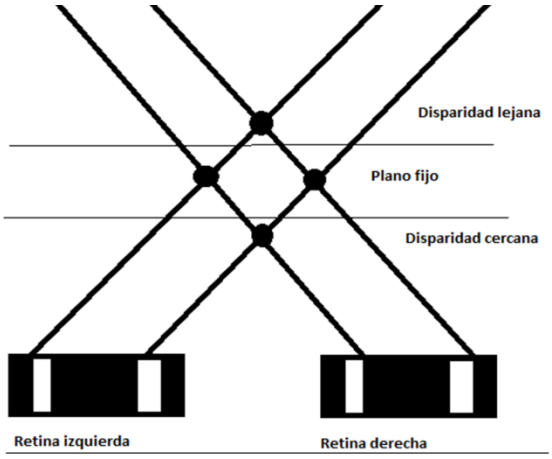


Figura 2.7. Filtro de disparidad en V2 [41].

V2 Superficies monoculares. En esta parte, se produce la percepción de superficies con base a un llenado de regiones que se acotan mediante los bordes. Para ello, la etapa de tejido delgado recibe señales de luz de los LGN desde V1 blobs. El llenado se modela a partir de una ecuación de difusión dada por:

$$ee\frac{d}{dt}F_{ijd}^{L/R} = -\alpha_L F_{ijd}^{L/R} + \sum_{(p,q)\in N_{ij}} \left(F_{pqd}^{L/R} - F_{ijd}^{L/R}\right)\phi_{pqijd} + x_{ijd}^{L/R} \tag{2.4}$$

donde ee<<1, α_L es una constante de caída, $F_{ijd}^{L/R}$ es el potencial de membrana, $x_{ijd}^{L/R}$ es la señal de luz de los LGN, ϕ_{pqijd} es un coeficiente de difusión que representa la inhibición de la difusión debido a las señales de los bordes. Este proceso ha sido muy documentado por diversas teorías y da una explicación a la separación de figuras y percepción de luz [41].

***V4 Superficies*.** La percepción en 3D es formada en el área V4. Esta etapa recibe señales de cada una de de los LGN vía V1 blobs y V2 tejido delgado. Recibe también los bordes de V2. Aquí se combinan las señales monoculares de luz en los diversos planos de profundidad correspondientes.

2.2.2 FORMOTION

FORMOTION o Forma-Movimiento, es un modelo que parte de diversas teorías que sostienen que el cerebro entre sus funciones realiza un proceso de segmentación e integración de movimiento partiendo de una serie de señales recibidas de forma local por los campos receptivos y va generando un modelo de percepción global. No obstante, un inconveniente en la visión que se da en los campos receptivos es el "problema de apertura", el cual consiste en que el movimiento de una línea visto desde una apertura circular pequeña es perceptualmente ambiguo. Por ejemplo, el movimiento paralelo a la dirección de una línea no se puede detectar por un solo campo receptivo; solo se puede detectar el movimiento perpendicular. Otro problema que surge en la detección de movimiento, es cuando múltiples objetos se mueven detrás de oclusiones en aperturas ambiguas, por lo que se puede tener un problema de percepción no real de movimiento [42]. Sin embargo, a pesar de ello, el sistema visual puede hacer segmentación integrando cada parte para hacer una percepción coherente de movimiento.

Los procesos de forma y movimiento se hacen a través de distintas interacciones entre las áreas de V1, V2, MT y MST (medio temporal superior). Para resolver y encadenar adecuadamente los problemas de integración de movimiento, el sistema visual utiliza patrones ambiguos que surgen de cambios de iluminación. A este proceso se le conoce como "seguimiento de características". Para dar solución al problema de apertura, se cree que el seguimiento de características se propaga gradualmente a través de la captura de señales de movimiento direccional en el área cortical MT. De acuerdo a muchos datos psicológicos, las características de seguimiento pueden propagarse gradualmente a través del espacio para capturar señales ambiguas.

En la Figura 2.8 se puede ver el modelo FORMOTION, el cual tiene dos procesos en paralelo. Por un lado, está la percepción de formas, por otro lado está la percepción de movimiento. En procesamiento de formas, se puede ver que se tienen seis etapas que

inician desde las partes de ON, OFF de los LGNs. Para separar las formas en 3D del fondo, la teoría FACADE (percepción de objetos con base a Forma, Color y Profundidad) propone un sistema que predice cómo los bordes de los objetos que generan oclusión son separados por profundidad de otras superficies incluyendo bordes intrínsecos y extrínsecos en V2. Esto genera una separación crucial en el modelo FORMOTION [43].

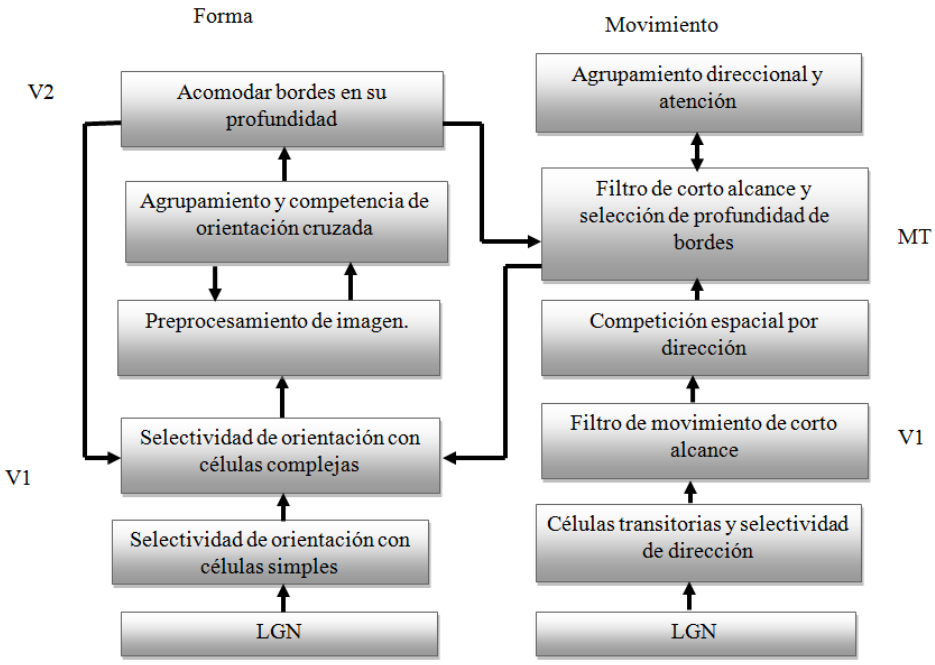


Figura 2.8. Pasos del procesamiento de 3D FORMOTION **[**41**]**.

La parte del modelo FORMOTION dedicada al análisis de forma es básicamente lo mismo que la detección de forma e integración de contornos del modelo LAMINART. Por tanto, el análisis de FORMOTION se realiza en esta tesis solo en la parte de análisis de movimiento. En las Figuras 2.9 y 2.10, se muestran los resultados de una simulación de los niveles de LGN las capas corticales de V1 del modelo FORMOTION; en la Figura 2.9 se observan los principales pasos del video *chopstick*[6] (un video utilizado en **[**43**]**) el cual tiene dentro de la escena dos líneas negras que se mueven en la dirección que marcan las flechas azules. Es decir, primero se van alejando en direcciones contrarias del cuadro 1 hasta el cuadro 30, luego se van acercando moviéndose en la misma dirección hasta el cuadro 100, para luego ir en direcciones contrarias hasta el cuadro 140. En la Figura 2.10 se pueden observar los resultados de las capas corticales de LGN y V1 implementadas con un

[6] Video disponible en http://cns.bu.edu/~juliaber/formotion.html

conjunto de ecuaciones definidas en variables discretas y son una variación del modelo original en [43]. Durante este experimento se intentó implementar el modelo original con el método de *Runge-Kutta* de cuarto orden, utilizando una *Workstation Dell Presicion* con procesador *XEON* de 4 núcleos y 4 Gb de memoria RAM. No obstante, cada iteración era procesada en alrededor de 3.85 segundos, y el ultimo nivel de V1 saturaba la memoria RAM. Por lo tanto, se realizó una versión discreta del modelo FORMOTION que parte desde las ecuaciones de los LGN hasta el último nivel de V1. Los resultados mostrados en la Figura 2.10a hasta 2.10j son los mismos que en el modelo continuo, las Figuras 2.10k y 2.10l muestran el resultado del nivel que no se pudo correr en el modelo continuo.

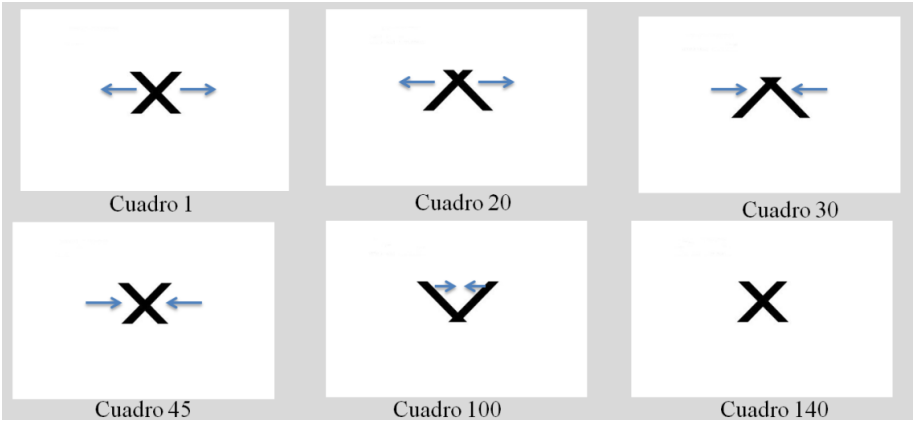


Figura 2.9. Ilusión de *chopstick*. Dos líneas que se mueven en las direcciones que indican las líneas azules. Al inicio las líneas negras se mueven en direcciones contrarias, luego a partir del cuadro 30, las líneas se empiezan a acercar. A partir del cuadro 45 las líneas se vuelven a alejar hasta el cuadro 100. Finalmente, se regresan a la posición original en el cuadro 145.

En cuanto al sistema de movimiento, éste se encuentra separado seis niveles, los cuales se describen a continuación, haciendo énfasis en los niveles 2,3 y 4. En [43] se puede analizar el modelo completo.

Nivel 1 LGN: Al igual que LAMINART, los LGN hacen un procesamiento de campos receptivos que habilitan el centro de la entrada y atenúan los alrededores. No obstante, en este caso, las células ON responden a los bordes anteriores, mientras que las células OFF a los bordes posteriores.

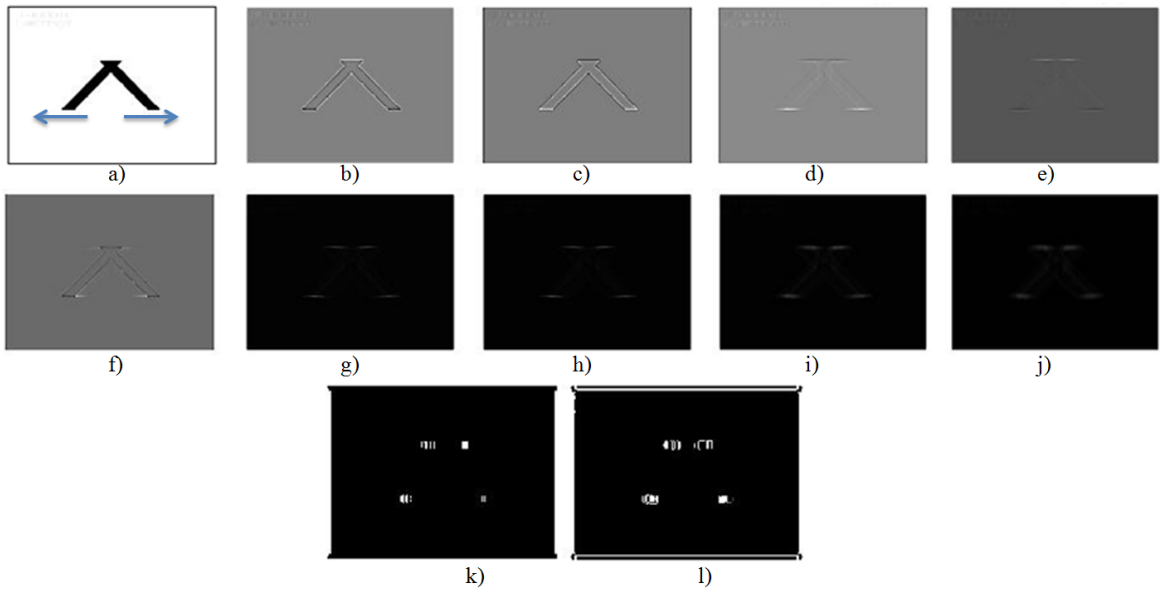


Figura 2.10. Resultado de implementar en forma parcial el modelo FORMOTION utilizando el video de *chopstick*. a). Cuadro de entrada. b). - c). Respuesta de Núcleos Laterales Geniculados. d). - e). Respuesta del potencial acumulado en las células transitorias f). Respuesta de células transitorias. g). Células Interneuronales. h). Células direccionales. i). - j). Nivel 3 en escalas *s=1* y *s=2*. k). - l). Respuesta del Nivel 4.

Nivel 2 Células Transitorias: Este nivel consiste en un conjunto de células transitorias direccionales y no direccionales. Las células transitorias no direccionales (CTND) responden a cambios de iluminación y realizan una acumulación de potencial para obtener patrones de movimiento como se ve en la Figura 2.10f. La entrada de las CTND es la respuesta ON y OFF de los LGN (Figuras 2.10b y 2.10c), y estas células tienen tres subniveles. En el primer subnivel la información de los bordes se inhibe, pero la información de las esquinas de los bordes se estimula. En el segundo subnivel existe una acumulación de potencial; de tal manera que en esta capa cortical la información de las esquinas se acumula formando un patrón temporal como se ve en las Figuras 10.d y 10.e, mientras que los bordes se inhiben. En el tercer nivel de las CTND los canales ON y OFF se agregan en una respuesta como se ve en la Figura 2.10f donde los bordes de los objetos quedan completamente inhibidas, mientras que el patrón de los bordes se inhibe. La siguiente etapa son las células interneuronales direccionales inhibidoras (CIDI), las cuales activan las células transitorias direccionales (CTD). Las CTD se estimulan cuando un estímulo se mueve en una dirección determinada; cada célula es selectiva a una dirección D, y si la salida de CTDN se mueve en esa dirección, entonces esta célula se estimula, de otro modo se inhibe. En las Figuras 2.10g y 2.10h se observa la respuesta de las CIDI y CTD sensibles a 180º.

Nivel 3 Filtro de movimiento de corto alcance. En esta parte se hace el procesamiento para solucionar el problema de apertura. En este nivel un conjunto de células simples cuya respuesta es un campo receptivo gaussiano generan una respuesta que estimula los patrones acumulados que definen el movimiento en una dirección determinada. Este nivel, analiza el resultado de las células direccionales desde varios niveles de escalabilidad y realizar un filtrado para integrar las señales espacio temporales del movimiento y descartar bordes de objetos. En las Figuras 2.10i y 2.10j se puede observar el resultado del filtro de corto alcance en dos escalas, las cuales varían en la amplitud del campo receptivo gaussiano.

Nivel 4 Competición espacial y de dirección oponente: Este nivel toma como entrada los resultados del nivel 3. En este nivel existe un conjunto de competiciones donde los patrones formados por el movimiento se estimulan y el resto de la información se inhibe. Como se observa en la Figura 2.10k y 2.10l, el resultado en las dos escalas contiene una serie de patrones que siguen el movimiento del objeto. Estos patrones siguen las trayectorias del movimiento causado por los objetos que aparecen en la escena.

Nivel 5 Selección de forma-movimiento*:* Esta parte incluye el procesamiento que se realiza en la etapa MT de la corteza. Para ello, ésta área recibe la salida de V1 de la capa 4 de movimiento y proyecciones de V2. Esto es para retroalimentar las señales de seguimiento de movimiento con el procesamiento de forma. De manera que en esta fase se asocian las señales de movimiento con las formas mapeadas en superficies que se encuentran a distintos niveles de profundidad. Por lo tanto, en este nivel se realiza en análisis de forma-movimiento (FORMOTION), el cual consiste en una selección de señales de movimiento en profundidad por sus bordes correspondientes.

Los bordes mapeados en superficie de profundidad obtenidos de V2 se relacionan con los patrones de movimiento y en consecuencia a los patrones de movimiento se asigna el nivel de profundidad y se detectan los posibles oclusores. Es decir, en este nivel los patrones de movimiento se asocian con los objetos que los causaron y sus niveles de profundidad.

Nivel 6 Agrupamiento direccional*:* El modelo de procesamiento no resuelve el problema de apertura. Para ello se tiene una etapa de agrupamiento direccional, la cual trabaja de la siguiente manera: Células con direcciones similares en MT son enviadas al nivel Medio

temporal Superior (MST) en forma agrupada. Luego, en MST se genera un proceso de competición direccional para determinar una dirección de movimiento ganadora, la cual retroalimenta a la capa MT para suprimir movimiento en otras direcciones.

En conclusión, se puede ver que el modelo FORMOTION obtiene los patrones de iluminación espacio-temporales que describen parte de la trayectoria del objeto en movimiento. Estos patrones de movimiento se relacionan con el objeto que los define y la profundidad. Mediante estas características se genera un proceso de percepción donde se encuentra la dirección del movimiento de un objeto.

2.2.3 Otras teorías basadas ART

Aparte de LAMINART y FORMOTION, existen varias teorías que explican los mecanismos de percepción dados por la corteza visual y otras áreas del cerebro. ARTSCAN y ARTSCENE [44] son dos teorías que tienen bases similares a las descritas anteriormente. El modelo ARTSCAN explica cómo la atención espacial y de objetos contribuyen en los procesos de aprendizaje. Este modelo se basa en la formación de representaciones de objetos mediante un proceso neuronal de atención y sostiene que en una escena con múltiples objetos, siempre habrá un objeto ganador que producirá una superficie de resonancia, para generar el proceso de aprendizaje. En la Figura 2.11 se expone un diagrama general de ARTSCAN, donde se muestra el procesamiento del modelo y su analogía con diversas áreas de la corteza visual [44]. Se puede observar que este modelo tiene una etapa de pre-procesamiento donde se detecta el objeto con base a la superficie y el contorno. A partir de ahí se realiza un proceso de atención espacial, mientras que por otro lado, se efectúa un proceso de reconocimiento del objeto. Otro esquema es el de ARTSCENE, y se trata de un modelo bioinspirado para clasificación de objetos. El modelo puede incrementar su aprendizaje y predecir la identidad de la escena mediante el clasificador default ARTMAP2. Además puede mejorar su proceso de atención selectiva de texturas [45].

2.3. Redes Neuronales Pulsantes

Actualmente, se asume que las neuronas en el sistema nervioso tienen una respuesta basada en pulsos. Numerosos modelos de oscilaciones y pulsos se han presentado desde mediados de 1980. En dichos modelos la información es procesada mediante la

sincronización de oscilaciones o pulsos. Los modelos de redes pulsantes que se reportaron durante la búsqueda que se basan en el funcionamiento de la corteza visual son:

- Redes Neuronales Oscilatorias.
- Redes Neuronales Integra y Dispara.

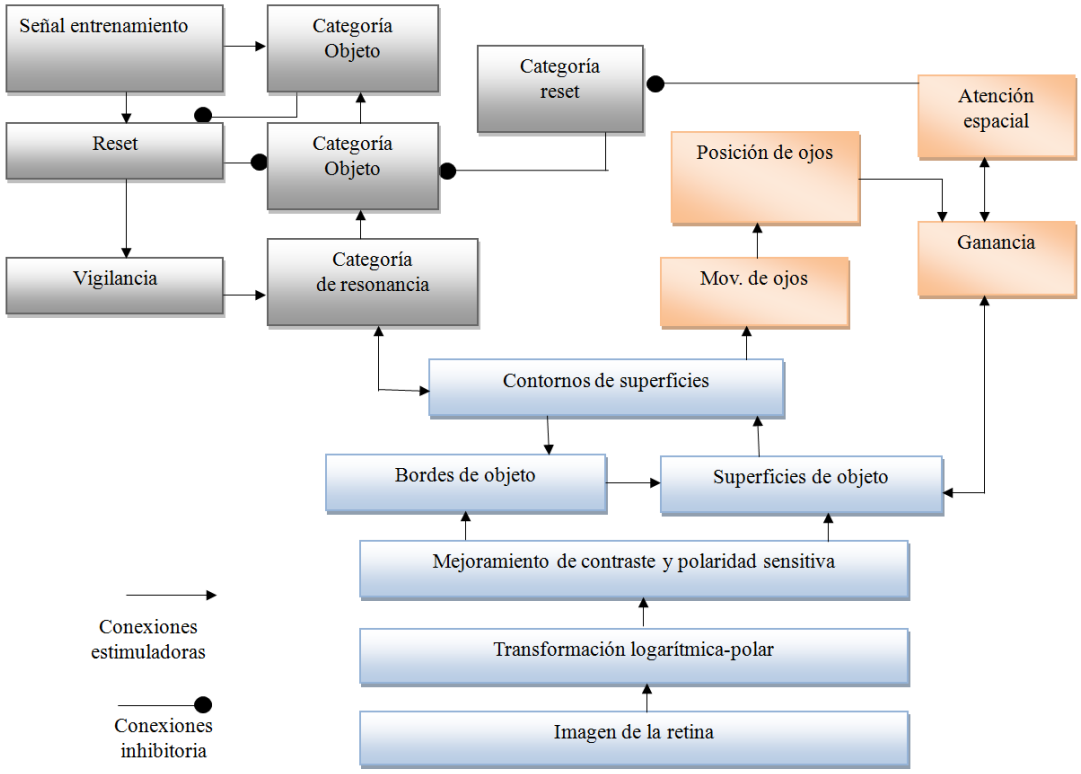


Figura 2.11. Modelo ARTSCAN **[**44**]**.

Un punto interesante es que los modelos pulsantes son RNA aplicadas principalmente para solucionar problemas en áreas como visión por computadora y procesamiento de imágenes, mientras que los modelos anteriores LAMNART, ARTSCAN, FORMOTION son modelos neurocomputacionales enfocados a describir el comportamiento de ciertas áreas de la corteza visual.

2.3.1 Redes Neuronales Oscilatorias

Este tipo de redes neuronales se enfocan a describir al sistema nervioso como un conjunto de oscilaciones acopladas que forman una dinámica temporal que va agrupando neuronas. En este caso, cada oscilador es determinado por dos variables que forman el estado de un oscilador acoplado, en el cual una variable estimula la oscilación mientras otra variable la inhibe. Cada oscilación representa la actividad de una neurona donde la unidad

estimuladora representa el potencial de membrana y la unidad inhibitoria representa el cambio del potencial que resulta de los canales iónicos en cada neurona **[**34**]**. Las redes neuronales oscilatorias se basan en la hipótesis que ciertas regiones del cerebro funcionan a través de correlación de oscilaciones, las cuales se disparan con estimulación sensorial. En la literatura analizada, se reportaron dos modelos de redes oscilatorias, las cuales son el modelo LEGION [46] (Red neuronal Oscilatoria Localmente estimulada y globalmente inhibida) y las redes basadas en oscilaciones de *Wilson-Cowan* [47].

Modelo LEGION. Evidencia experimental muestra que existen oscilaciones neurales en la corteza visual que hacen posible los mecanismos para detectar características en una escena. Las neuronas forman oscilaciones que parten de potenciales de acción que responden a características como color, orientación y movimiento. Además, estudios han dado evidencia que estímulos con oscilaciones de alrededor de 40 Hz en la corteza visual generan un comportamiento sincronizado agrupándose de manera espacial. Esto sugiere que las células funcionan como detectores de características visuales haciendo una correlación en sus actividades de disparo. Con base a estas oscilaciones, LEGION fue propuesto por Terman y Wang en [46] como un modelo computacional bioinspirado para análisis de imágenes. Se trata de una red de oscilaciones construidas por un excitador x_L y un inhibidor y_L. En la Figura 2.12a se observa cómo interactúa este modelo, donde x_L es el excitador, y_L es el inhibidor, I_L es la entrada y *SS* es la respuesta del resto de la red. Aquí, la unidad x_L envía una excitación a la unidad y_L, la cual responde enviando una inhibición. Cuando un estímulo externo I_L se aplica continuamente, este lazo de retroalimentación causa una oscilación. Para el caso de esta red, la arquitectura se basa en una unidad y_L que va a inhibir a un vecindario de excitadores x_L, creando un modelo basado en una red de excitadores con un inhibidor global, tal como se observa en la Figura 2.12b. En las ecuaciones 2.5 y 2.6 se observa la dinámica con la que funciona la red.

$$\dot{x}_{Li} = 3x_{Li} - x_{Li}^{3} + 2 - y_{Li} + \rho_g + I_{Li} H\left(p_i + e^{-\alpha t} - \theta_{\lg}\right) + SS_i \tag{2.5}$$

$$\dot{y}_{Li} = e \cdot \left(\gamma \cdot \left(1 + \tanh\left(x_{Li} / \xi\right)\right) - y_{Li}\right) \tag{2.6}$$

Este modelo es sustentado mediante los osciladores de *Van Der Pol* [46], y la dinámica del excitador x_L se define en la ecuación 2.5 como una ecuación diferencial de primer orden con un comportamiento cúbico. Aquí el término y_{Li} representa el inhibidor de

la unidad y_L, y *SSi* representa el acoplamiento con los osciladores vecinos y el inhibidor global. El excitador *i*

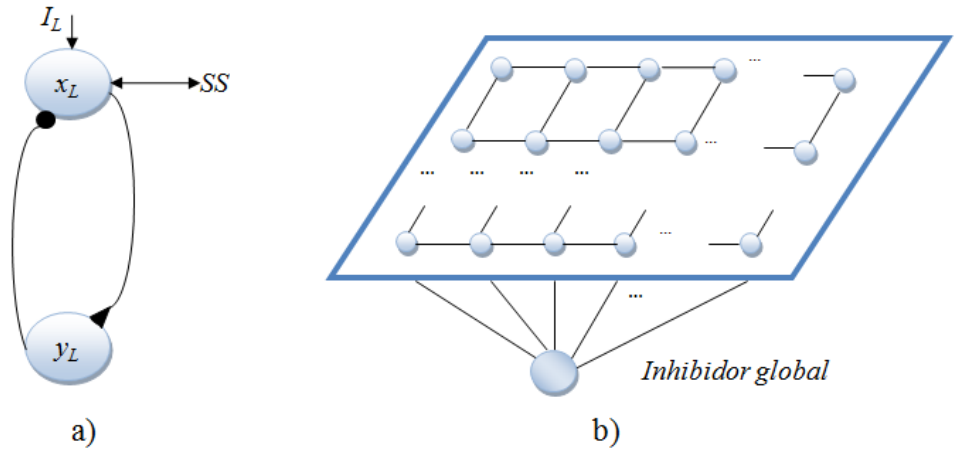


Figura 2.12. Modelo LEGION. a). Un oscilador simple. b). Una red LEGION bidimensional.

se estimula si $I_{Li} > 0$. La función de *Heaviside* se define como $H(a)=1$ sí $a \geq 0$ y $H(a)=0$ sí $a<0$. La función H determina un comportamiento oscilatorio multiplicándose por I_{Li}. La variable p_i es el potencial lateral del oscilador y se utiliza para suprimir regiones ruidosas. El parámetro θ se coloca en el rango de $(0,1)$, se utiliza como un umbral para p_i y se tiene también un término exponencial que cae según ξ. El oscilador lateral que excede θ_{lg} se refiere como el líder, es decir, define la frecuencia para un conjunto de osciladores. El parámetro ρ_g es la amplitud de ruido gaussiano. El comportamiento de y_L es definido en la ecuación 2.6 donde se observa una función sigmoidal, γ, e son parámetros, donde $\gamma = 2$, $0<e<<1$ y además, determinan las escalas de tiempo del oscilador. La dinámica de la red se muestra en el diagrama de fase de la Figura 2.13a, donde el resultado es una órbita periódica que define un ciclo límite que forma la oscilación que muestra la Figura 2.13b. Además, las trayectorias nulas (dinámica definida en las ecuaciones 2.5 y 2.6 cuando $\dot{x}_{Li}$ e $\dot{y}_{Li}$ son cero) definen los puntos en los cuales la oscilación se inhibe. Se puede observar en la ecuación 2.5 que I_L y *SS* tienen un efecto de desfasamiento o ganancia en la ecuación cúbica; la posición de la ecuación cúbica con respecto a la sigmoidal define dos estados para un oscilador. El primero es un estado de habilitación donde los puntos de la sigmoide y la cúbica se igualan con un punto del ciclo límite. El estado excitable ocurre cuando la ecuación cúbica cambia desde el punto de separación de la cúbica y el ciclo límite hasta donde coincida con la sigmoidal. Esta dinámica implica que con el tiempo un bloque de

oscilaciones correspondientes a una región en una imagen tienden a sincronizarse, mientras dos oscilaciones correspondientes a dos regiones, tienden a desincronizarse.

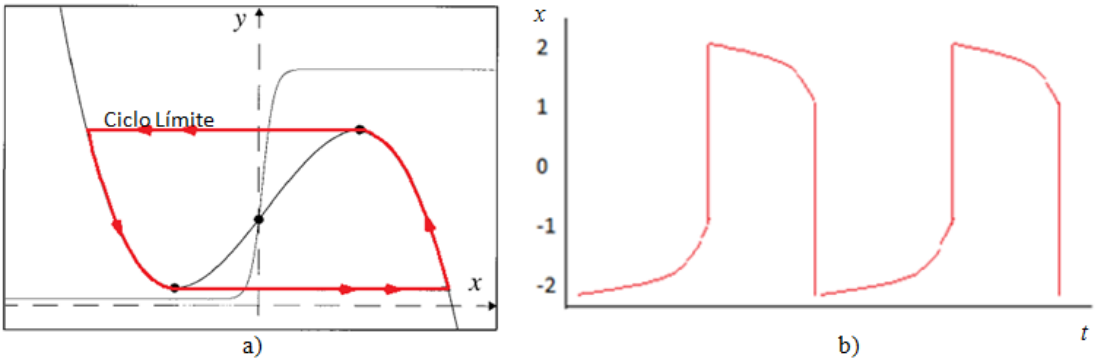


Figura 2.13. Dinámica del modelo LEGION. a). Diagrama de fase. b). Respuesta del modelo.

En la literatura analizada, se observó que la mayor parte de las aplicaciones son en segmentación en imágenes y se sostiene la hipótesis de que las neuronas tienen un comportamiento oscilatorio que se va sincronizando para segmentar regiones visuales que representen objetos coherentes de forma tal que dicho agrupamiento se puede comparar con los principios de la *Gestalt*. Las aplicaciones que tiene LEGION es en segmentación de imágenes en MRI [48**,**49**]**, segmentación de caminos en imágenes [50] y segmentación de texturas [51]. El modelo LEGION también se encontró en diversas implementaciones en hardware; en [52] y en [53] se utilizó un FPGA para segmentación de imágenes, mientras que en [54] y [55] se implementó este modelo en procesadores CMOS y VLSI (*Very Large-Scale Integration*). En [56] se observa una aplicación donde el modelo LEGION es utilizado para resolver un problema común en percepción conocido como el problema de la espiral, el cual consiste en una espiral simple conectada y dos espirales desconectadas, similar a lo que se observa en la Fig. 2.14 donde se ven ambas figuras y el procesamiento dado por LEGION. Aquí se sostiene y se comprueba la hipótesis de que la correlación de oscilaciones tiene una dinámica que permite resolver problemas de Percepción Visual de manera similar a los procesos perceptuales al nivel del sistema nervioso.

En resumen, el modelo LEGION genera una oscilación que representa una región dentro de una imagen digital. Este modelo es exitoso en imágenes a escala de grises y lógicas, además de que segmenta partiendo las imágenes en más de una región y hoy en día existen muchos trabajos en los que se realizan implementaciones en Hardware.

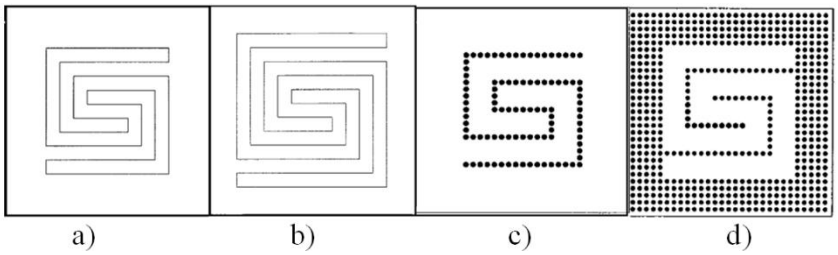


Figura 2.14. Problema de la espiral **[**56**]**. a). Espiral simple conectada. b). Espiral doble desconectada. c). - d). Procesamiento de las espirales con LEGION respectivamente de a) y b).

Una variante de LEGION presentada en [57] es la red de conexiones difusivas (el estado de cada neurona influye en el estado de neuronas vecinas). Dicha red consiste de un conjunto de osciladores con varios inhibidores. Estos osciladores se conectan para agregar una serie de conexiones difusivas para generar grupos perceptuales. Para describir estas conexiones, primero se define el siguiente espacio de estados:

$$\dot{\delta}_i = \begin{bmatrix} \dot{v}_i \\ \dot{\omega}_i \end{bmatrix} \tag{2.7}$$

donde v_i es el potencial de membrana del oscilador, ω_i es una variable de estado que representa el voltaje de compuerta. La notación de este modelo está respecto a [57]. Este vector de estado del oscilador se expresa como $\dot{\delta}_i = f(\delta_i, t)$. Una red compuesta de N_δ osciladores se define como:

$$\dot{\delta}_i = f\left(\delta_i, t\right) + \sum_{i \neq j} \nabla_{ij} \left(x_j - x_i\right), \qquad i = 1, ..., N_\delta \tag{2.8}$$

donde ∇_{ij} es el peso que representa la conexión entre las neuronas i,j. En el caso de que los osciladores ∂_i y ∂_j tengan la misma respuesta, entonces estos se van a sincronizar. De acuerdo a [57], esta conexión difusiva tiene ciertas similitudes con la forma de agrupar los elementos en la teoría de la *Gestalt*. Para dar más similitud al modelo difusivo con el principio de agrupamiento, de acuerdo a [57], la evidencia psicofísica muestra que en los modelos oscilatorios existe una sintonización que se puede aproximar a un comportamiento gaussiano, por lo que los pesos ∇_{ij} se definen como gaussianas dadas por:

$$\nabla_{ij} = e^{\left(\frac{(x_i - x_j)^2}{-\sigma_\nabla}\right)} \tag{2.9}$$

donde x_i, x_j son los estímulos de los osciladores, σ_∇ es el radio de difusión de cada oscilador. El modelo en [57] es jerárquico para dar mayor plausibilidad con el funcionamiento de la corteza visual, de manera que en este modelo existen otras redes difusivas que se conectan con la primera red mediante conexiones hacia arriba (*bottom-up*) y de regreso (*top-down*). Esta red se prueba para dos de las principales tareas que realiza la corteza visual, las cuales son integración de contornos y segmentación de regiones. En ambos casos, este modelo oscilatorio de conexiones difusivas puede sustentar principios de la *Gestalt*; en el caso de la integración de contornos, se puede sustentar el principio de la *Gestalt* de continudad y el caso de la segmentación de regiones sustenta el principio de constancia de color de la *Gestalt*. Para integración de contornos y para color el modelo es muy similar, pero cambian los argumentos de la exponencial gaussiana de la ecuación 2.9.

En resumen, la red de oscilaciones difusivas es un conjunto de osciladores que se sincronizan con base a los principios del agrupamiento perceptual. Para lograr esta forma de agrupamiento, se tiene un conjunto de redes basadas en vectores de estados con un conjunto de pesos con comportamiento gaussiano. Las redes difusivas al igual que muchos métodos perceptuales tienen conexiones hacia arriba y de regreso. Los vectores de estado son oscilaciones basadas en el modelo de *Van Der Pol* como la red LEGION.

Modelo de *Wilson-Cowan*. En Neurociencia Computacional, el modelo de *Wilson-Cowan* [**47**] describe las interacciones en grandes poblaciones de neuronas. Con base a este modelo, se han desarrollado diversos modelos para analizar diversos aspectos en la percepción de objetos. Uno de estos se refiere a que la representación de objetos requiere de varios procesos cognitivos que ocurren en paralelo y en forma distribuida en el cerebro; de manera que varias características de objetos son procesadas en distintas áreas corticales.

Basándose en osciladores de *Wilson-Cowan*, se ha planteado la hipótesis de correlación temporal, la cual postula que grupos neuronales representan diferentes aspectos de un mismo objeto a través de sincronización de células en frecuencias de 30-100 Hz. Esta sincronización se ha visto en el procesamiento de cortezas sensoriales como la visual, somato-sensorial, auditiva y olfativa. Además, la sincronización neural tiene un rol importante en percepción sensorial y en tareas cognitivas [47]. El reconocimiento múltiple

de objetos es similar a cualquier proceso de percepción, pero con dos diferencias principales:

- Los objetos son considerados una colección de características.
- El reconocimiento se hace a partir de conocimiento previo y similaridad. El principio de similaridad se implementa a través de mapas topológicos.

Existen muchos modelos que se sustentan en los principios de agrupamiento de la teoría de la *Gestalt*. Estos se basan en múltiples capas corticales y áreas para procesar separadamente distintas señales de los patrones visuales de entrada, de forma similar a la anatomía de la corteza visual.

La red basada en oscilaciones de *Wilson-Cowan* trata de simular los procesos de segmentación en altos niveles cognitivos para hacer una separación simultánea de objetos y su reconocimiento en una sola capa de procesamiento mediante agrupamiento de características fundamentales basándose en las reglas de la *Gestalt* de similaridad y conocimiento previo [47]. Esta red contiene varias áreas corticales dedicadas a representar una característica específica de un objeto. La regla de similaridad se realiza mediante conexiones entre neuronas con formas de "*mexican hat*". Cada área está compuesta de M_1xM_2 osciladores. Por lo que el número total de osciladores es:

$$N_o = L_o \cdot M_1 \text{x} M_2 \tag{2.10}$$

El oscilador puede no tener actividad si no recibe suficiente estimulación, o puede oscilar en la banda gamma si se estimula lo suficiente (30-100 Hz). En esta red los objetos son representados por grupos de neuronas con L_o características. Para esto, se asume que cada atributo no se presenta de manera inmediata en las entradas sensoriales, pero ha sido extraído de un procesamiento previo en el neocortex[7]. Los grupos neuronales en las mismas áreas tienen conexiones con excitación lateral y sinapsis inhibitorias. Las conexiones laterales tienen la forma del "*mexican hat*". El modelo de cada excitador se puede observar en la Figura 2.15. Aquí, las flechas representan las conexiones excitatorias y las líneas con puntos representan conexiones inhibitorias. En este modelo, x_w representa la actividad de la población excitatoria y_w la actividad de la población inhibitoria. Los términos α_w, β_w

[7]Neocortex es la parte más evolucionada de la corteza cerebral que está más asociada a las capacidades cognitivas.

representan las fuerzas de enlace entre las poblaciones, I_w es el estímulo externo y v_w representa el ruido aleatorio. Z_w representa el inhibidor global, L^{EX}, L^{IN} las sinapsis laterales inhibitorias y excitatorias en la misma área. W es la sinapsis entre el oscilador y el resto de las áreas. Referente a la literatura analizada, son pocos los trabajos que se han realizado utilizando esta arquitectura.

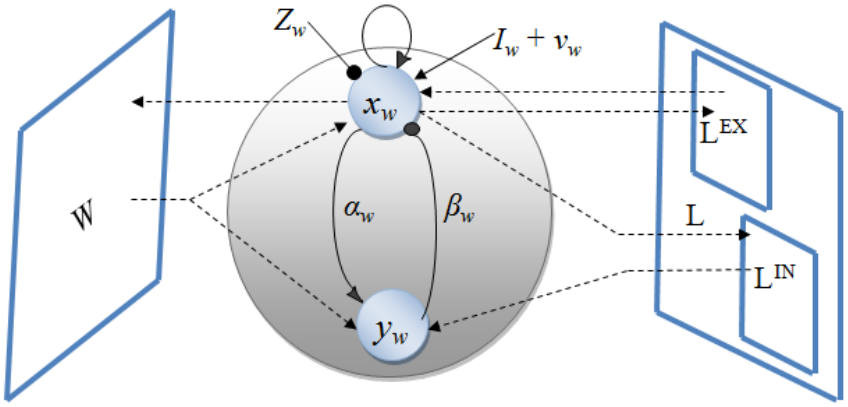


Figura 2.15. Estructura de cada oscilador y sus conexiones sinápticas **[**47**]**.

Los artículos publicados con esta red y los resultados reportados son en simulación, es decir; son imágenes creadas para pruebas, no de escenas reales. Esto se puede deber al alto grado de complejidad que tiene el poder implementar las oscilaciones *Wilson-Cowan* [58]. No obstante, resulta interesante la forma en cómo hace el procesamiento de las imágenes ya que las procesa como las partes altas de la corteza cerebral.

2.3.2 Redes Integra y Dispara (SNN)

En este tipo de neuronas, el potencial de membrana es representado por una variable cuya dinámica genera un impulso cuando el potencial excede cierto umbral [34]. Una formulación típica de este modelo es:

$$Cap\frac{dv_c}{dT} = i_c(t) - \frac{v_c}{\mathrm{Re}} \tag{2.11}$$

donde v_c es el potencial de membrana, *Cap* es la capacitancia, *Re* es la resistencia e $i_c(t)$ es la corriente de entrada. Como se puede notar, esta ecuación describe una neurona como un capacitor que funciona con base al efecto de $i_c(t)$, mientras que $-v_c/R$ es el término de retardo para la formación del impulso. En [59] y [60] se considera que las SNN forman la tercera generación de RNA [59]. La primera generación de RNA se refiere a los modelos

con neuronas de función de activación binaria tales como la red perceptrón o el modelo ADALINE [59]. La segunda generación se refiere a las RNA de función de activación continua como la sigmoidal, la función lineal saturada o las de función de base radial. La naturaleza pulsante de las neuronas biológicas ha sido fuente de inspiración de distintos modelos de RNAs. Dentro de estos modelos, existe una serie de neuronas artificiales cuya salida es un conjunto de impulsos similares a los que se observan en la Figura 2.16. La dinámica de las SNN se basa en evidencia experimental que muestra que neuronas de sistemas biológicos procesan la información en el dominio del tiempo mediante potenciales de acción que en ocasiones también se les conoce como espigas (*spikings*), similares a lo que se observa en la Figura 2.16a.

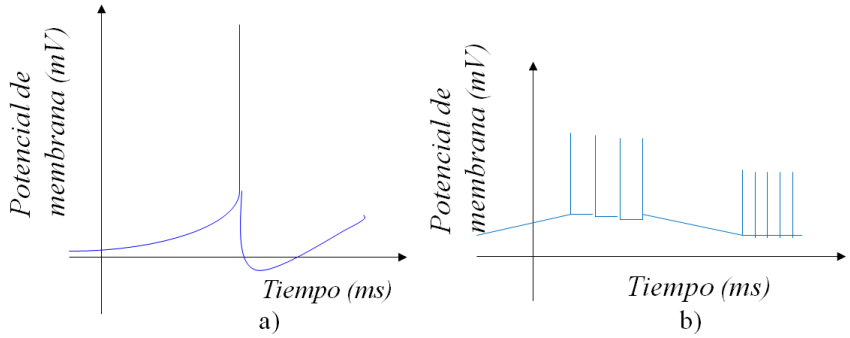


Figura 2.16. Impulsos. a). Espiga o pulso en sinapsis. b). Tren de pulsos (espigas) a la salida de neurona.

En general, las SNN tienen muchos modelos, uno de los modelos más viejos es el de *Hodgkin-Huxley* [61], el cual se trata de un sistema de ecuaciones diferenciales con la siguiente forma.

$$\frac{dv_c}{dt} = 0.4v_c^2 + 5v_c + 140 - u_c + I_c \tag{2.12}$$

$$\frac{du_c}{dt} = a\left(bv_c - u_c\right) \tag{2.13}$$

con un re-inicio después de cada impulso dado por: Si $v_c \geq 30\ mV$, entonces $\begin{cases} v_c \leftarrow c \\ u_c \leftarrow u_c + d \end{cases}$

Aquí v_c representa el potencial de membrana y u_c es una variable de recuperación [61], a,b,c,d son parámetros dimensionales. Típicamente, los valores de los parámetros son $a=0.02$, $b= 0.2$, $c= -0.65\ mV$ y $d = 2$. I_c se refiere a una corriente sináptica. Existen varios modelos de potenciales de acción. En general, dichos modelos tienen la siguiente forma:

$$\dot{v}_c = f(u_c) - u_c + I_c \tag{2.14}$$

$$\dot{u}_c = a(bv_c - u_c) \tag{2.15}$$

La cual tiene como objetivo generar como salidas trenes de impulsos. Existen muchos modelos dinámicos de trenes de impulsos, aparte del modelo de *Hodgkin-Huxley*, están los modelos de *Connor-Stephens* y el de *McGregor* [61]. Además de los múltiples modelos dinámicos de neuronas de impulsos, también se encontraron mecanismos de aprendizaje no supervisados para SNNs. El más común es el STDP (*Spike Timing Dependent Plasticity*) y se trata de un modelo inspirado en un mecanismo fisiológico que regula la activación de pulsos en las sinapsis. Este mecanismo genera un cambio en los pesos sinápticos dado por:

$$\Delta\omega_{ij} = \begin{cases} a^{+}\exp\left(\dfrac{t_j - t_i}{\tau^{+}}\right) & if \quad t_j - t_i \le 0 \quad (LTP) \\ a^{-}\exp\left(-\dfrac{t_j - t_i}{\tau^{-}}\right) & if \quad t_j - t_i > 0 \quad (LTD) \end{cases} \tag{2.16}$$

Donde t_j es el tiempo presináptico, t_i es el tiempo postsináptico, a representa la amplitud del impulso, LTP significa *Long term Potentiation* y LTD *Long Term Depression*. $\tau+$ y $\tau-$ son constantes que forman las aproximaciones sinápticas [62].

Referente a la literatura analizada, fue común encontrar trabajos donde se utilizaban las neuronas integra y dispara en alguna parte de un algoritmo o método para segmentación en imágenes y video. En este análisis, se pudo notar que no hay una arquitectura de red específica que utilice estos modelos en los trabajos estudiados; se plantean varios tipos de SNN, donde lo común es el uso de neuronas integra y dispara artificiales inspiradas a partir de los potenciales de acción y estas neuronas se plantean en diferentes tipos de redes. Existen diferentes redes que tienen algunas aplicaciones como se puede ver en [63] donde se documenta un análisis del estado del arte de las SNN, en el cual se estudian diversas arquitecturas de SNN, entre las que destacan la *Echo State Network* (ESN) y la *Liquid State Machines* (LSM), ya que estas han sido analizadas para el desarrollo de métodos de predicción, agrupamiento y análisis de señales. Adicionalmente, existen otras arquitecturas como la *Spiking Convolutional Neural Network* [64] o los mapas neuromórficos [65], las cuales han tenido ciertas aplicaciones en distintas tareas del procesamiento de imágenes.

Otra SNN ampliamente utilizada es la Red Neuronal Pulsoacoplada (PCNN), la cual es una red compuesta de varias neuronas artificiales propuesta por *Eckhorn* a partir de

modelos de la corteza visual de mamíferos. Luego se propusieron posteriores modificaciones por *Johnson*, *Ryback* y *Parodi*. En [66] se puede apreciar que esta red ha sido ampliamente utilizada en aplicaciones relacionadas con procesamiento de imágenes, segmentación y visión artificial. Por tanto, esta es una de las SNN más utilizadas en la literatura.

La arquitectura de la PCNN tiene tres módulos: el árbol de dendritas, el de encadenamiento y el generador de pulsos, tal como se muestra en la figura 2.17.

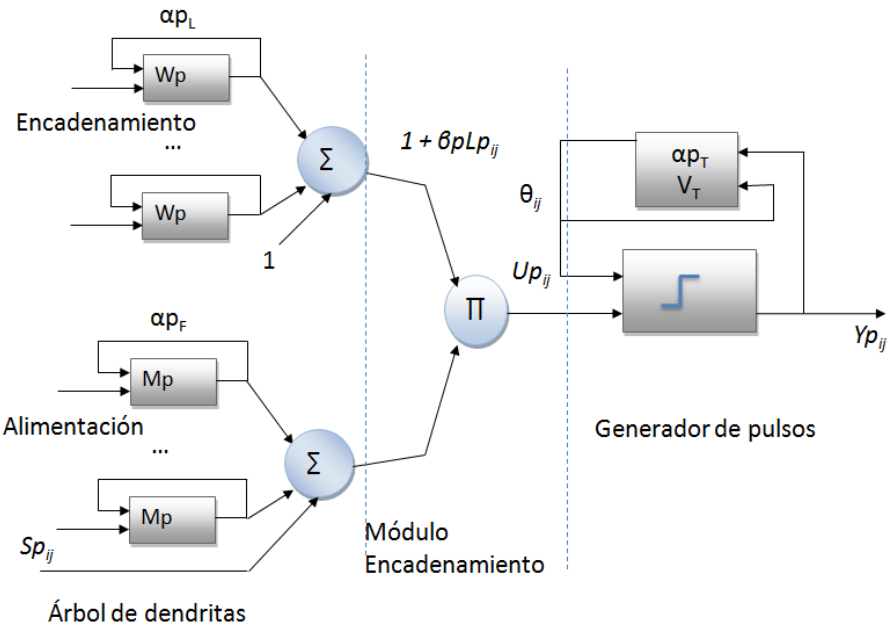


Figura 2.17. Arquitectura de la PCNN [67].

El árbol de dendritas está formado por el módulo de alimentación y el de encadenamiento. La neurona recibe la respuesta de neuronas vecinas del módulo de dendritas [67]. La entrada se obtiene por el módulo de alimentación. El generador de pulso compara la actividad interna *Up(t)* con un umbral dinámico para decidir si la neurona se dispara o no. El módulo de alimentación está definida por:

$$Fp(i,j)^{n_p} = Sp(i,j) + Fp(i,j)^{n_p-1} e^{-\alpha p_F} + Vp_F Yp_{ij}(i,j)^{n_p-1} * Wp \qquad (2.17)$$

donde *Fp(i,j)* es la respuesta del módulo de alimentación, *Sp(i,j)* es el pixel de entrada, αp_F es la constante de tiempo del filtro en la región de alimentación, Vp_F es un potencial, *Yp(i,j)* es la salida en la iteración n_p, *Wp* es el *kernel* de alimentación y n_p el índice de iteraciones de la PCNN. Se observa que *Wp* hace una convolución con la salida anterior. El módulo de encadenamiento está definido por:

$$Lp(i,j)^{n_p} = Lp(i,j)^{n_p-1} e^{-\alpha p_L} + Vp_L Yp(i,j)^{n_p-1} * Mp \tag{2.18}$$

donde αp_L es la constante de tiempo del filtro de encadenamiento, Vp_L es un potencial de encadenamiento y *Mp* es el kernel de encadenamiento. El generador de pulsos compara la actividad interna *Up(t*) con un umbral dinámico que decide si la neurona se dispara o no. Esta actividad está definida por:

$$Up(i,j)^{n_p} = Fp(i,j)^{n_p}\left[1+\beta pLp(i,j)^{n_p}\right] \tag{2.19}$$

$$\theta p(i,j)^{n_p} = \theta p(i,j)^{n_p-1} e^{-\alpha p_\theta} + Vp_\theta Yp(i,j)^{n_p-1} \tag{2.20}$$

$$Yp(i,j)^{n_p} = \begin{cases} 1 & Si \quad Up(i,j)^{n_p} > \theta p(i,j)^{n_p} \\ 0 & otra \quad forma \end{cases} \tag{2.21}$$

donde βp es el coeficiente de encadenamiento, αp_θ es la constante de tiempo del filtro de umbral y Vp_θ es la ganancia de umbral [67]. Esta red genera una salida pulsante a través de varias iteraciones que se interpreta como información binaria que describe características de la entrada. Al tener como entrada una imagen, la salida en cada iteración es una imagen binarizada que agrupa los pixeles con características comunes en la imagen de entrada.

En la Figura 2.18 se puede ver un ejemplo del funcionamiento de la PCNN, donde hace una segmentación de imágenes. Como se mencionó anteriormente, cada neurona artificial en la PCNN tiene como entrada la información de un pixel y genera en el tiempo un tren de pulsos que se pueden sincronizar para representar formar grupos de neuronas. Además, en la Figura 2.18, se muestra la salida de tres neuronas durante 140 iteraciones; la neurona 1 tiene como entrada un pixel que pertenece a una región blanca, la neurona 2 tiene como entrada un pixel de fondo negro y la neurona 3 una entrada que pertenece a un borde. Como se puede apreciar, las neuronas 1,2 y 3 se disparan en iteraciones diferentes, simulando el efecto de la sincronización de los potenciales de acción ante un estímulo de entrada. El resto de las neuronas tienden a dispararse de forma similar, de manera que se pueden formar regiones como se ve en la Figura 2.19a, donde se puede ver una imagen a escala de grises, la cual es la entrada a una PCNN. En esta red, cada pixel de la imagen es *Sp(i,j)*, de forma que cada pixel está conectado a una neurona en la PCNN. Por tanto, $Yp(i,j)^{n_p}$ forma una imagen binarizada en cada iteración. Si la salida de una neurona es cero

en una iteración, esta no pulsa, si es uno entonces genera un pulso en dicha iteración; en la Figura 2.19b se observa $Yp(i,j)^{n_p}$ en $n_p=14$.

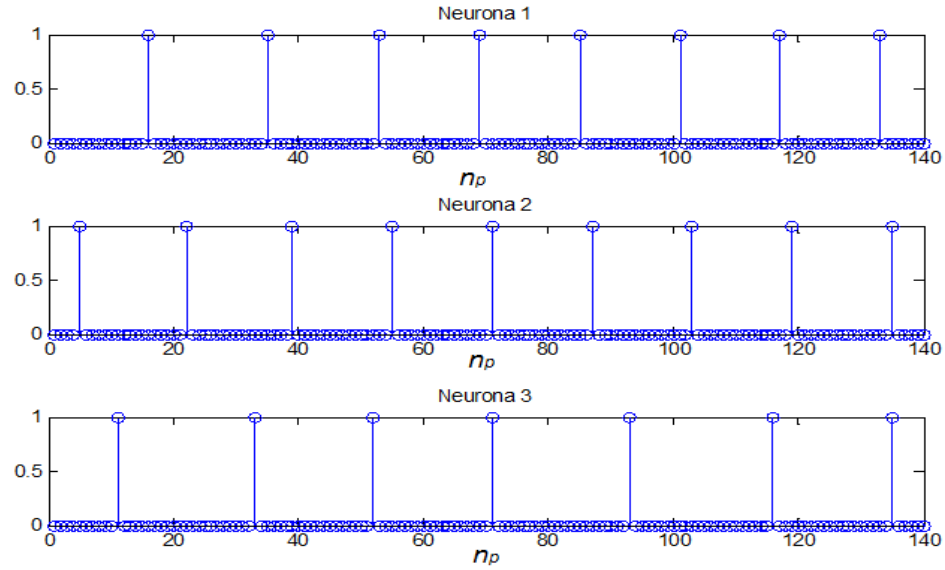


Figura 2.18. Trenes de pulsos de la PCNN.

Figura 2.19. Análisis de una PCNN. a). Imagen original. b). Salida de la red en la iteración 40. c). Serie de tiempo utilizando la ecuación 2.22.

Para codificar la información de la PCNN, se puede analizar por neurona el comportamiento de los pulsos. Una forma es las series de tiempo [66]. En este caso, se suma el resultado de todos los pulsos de la siguiente forma:

$$SYp(x,y)=\sum_{n=1}^{N_p} Yp(x,y)^{n_p} \tag{2.22}$$

donde $SYp(x,y)$ es una imagen que contiene la sumatoria de todas las salidas de la PCNN en las iteraciones de $n_p=0,\ldots,N_p$. En el caso de la imagen de la Figura 2.19a, si se utiliza la serie de tiempo dada por la ecuación 2.22 en $n_p=40$, la salida obtenida se muestra en la Figura 2.19c, donde se observa que la sumatoria de pulsos forma tres regiones, que segmentan de manera coherente los niveles de grises que se ven en la imagen original. Esta

segmentación es debida a la sincronización que tienen los pulsos de la red, la cual es similar a los mecanismos de segmentación de redes pulsantes como los modelos oscilatorios *LEGION*, *Wilson-Cowan* y diversas *SNN*.

2.4. Redes y Modelos Auto-organizados

Aunque existen diversos modelos inspirados en el comportamiento del sistema nervioso y el cerebro, estos modelos están muy lejos de imitar por completo dichos sistemas biológicos. Una razón es que el cerebro tiene aproximadamente $1x10^{11}$ neuronas con $1x10^{14}$ sinapsis, mientras que un sistema de cómputo tiene aproximadamente $1x10^{8}$ transistores [34]. Por tanto, una forma de llegar a desarrollar neuromodelos computacionales es basándose en el comportamiento de capas corticales y grupos de neuronas, en vez de basarse directamente en la dinámica de cada neurona. Un esquema que explica de forma global el comportamiento de neuronas es la auto-asociatividad de pesos; esto es, los pesos de distintas neuronas se van organizando en función de los estímulos sensoriales que las células aferentes trasmiten a las capas corticales. Aunado a esto, la corteza visual es una parte del cerebro cuyo funcionamiento se puede explicar en función de este concepto, ya que desde 1960s, Hubel, Wisel y sus colegas, realizaron numerosos experimentos que demostraban como se cambiaba la organización en la corteza visual en animales al cambiar el entorno visual [34]. Por ejemplo, si los ojos de un gato eran suturados, la corteza visual se volvía desorganizada al punto que el gato podía volverse ciego aun cuando sus ojos estuvieran sanos. Si el gato era enviado a un entorno determinado, la corteza visual del animal se organizaba de forma que solo reconocía patrones similares a dicho entorno. Además, existen estudios donde se analiza como la corteza visual de personas ciegas es utilizada en tareas auditivas, estímulos táctiles o lectura braille [68]. De esta forma, a través de diversos experimentos y evidencias, se ha visto que la corteza visual y otras cortezas sensoriales, tienen un funcionamiento que se modela a partir del estímulo de las entradas. Existen muchos modelos basados en RNA para formalizar este proceso de organización de pesos a través de estímulos sensoriales, un modelo es la red auto-organizada de Kohonen vista en la sección 1.2. Esta red ha sido muy estudiada para imitar mecanismos de la corteza visual como en **[**69**,**70], donde los autores desarrollaron un sistema basado en SOM para simular movimientos sacádicos; en [69], una SOM para solucionar el problema de selectividad de puntos, en [70] se utilizó para modelar

los receptores de la corteza visual. Alessio Plebe *et al*. [71] desarrollaron un sistema inspirado en los campos receptivos y otras áreas de la corteza visual basadas en SOM para reconocer objetos en distintas orientaciones. En [72], se documenta un modelo que proyecta señales de color y luminancia por koniocélulas (células que se encuentran en los LGN, se definirán en el capítulo V), donde la SOM se utiliza para simular el proceso cortical. Por otra parte, la SOM ha sido muy utilizada para proponer diversas soluciones neuroinspiradas para aplicaciones en el área de procesamiento de imágenes y visión por computadora, como en **[**73**,**74**,**75**,**76] donde diversos esquemas basados en filtros Gabor y SOM se propusieron para reconocimiento de señales de tránsito **[**73**]**, segmentación de textura [74], en [75] para agrupar imágenes en bases de datos basándose en características visuales y en [76] para percepción de movimiento.

Adicionalmente, existen diversos modelos diseñados para explicar el funcionamiento de ciertas capas de la corteza visual apoyándose en la hipótesis de que esta área del cerebro funciona bajo auto-organización de pesos basada en estímulos visuales. Ejemplo de esto es la familia de modelos LISSOM (Modelo Auto-organizado Sinérgicamente y Lateralmente Conectado), la cual se trata de un conjunto de modelos auto-organizados que se desarrollan a partir de varias teorías del funcionamiento de la corteza visual [34]. Bednar, Choe, Miikkulainen, y Sirosh **[**34**]** desarrollaron estos modelos basándose en la autoasociatividad de los pesos y células simples que se encuentran en la corteza visual. LISSOM se inició como una extensión de la SOM biológicamente más plausible y con varios tipos de conexiones laterales entre neuronas. Además, se inspira principalmente en las regiones de V1, LGN, la retina y el funcionamiento se basa en la simulación de neuronas corticales de conexiones aferentes que van adaptando conexiones mediante una actividad correlacionada, generando una estructura auto-organizada donde los pesos de las conexiones aferentes forman un mapeo de la señal de entrada y las conexiones laterales regulan la actividad neuronal.

2.4.1 Mapa Auto-organizado de Kohonen Retinotópico

El punto de partida para el análisis de los modelos LISSOM es conocer la forma en cómo SOM realiza mapas topológicos para explicar el funcionamiento de la corteza visual. Para ello, en [34] se parte de una variante de la SOM que se denomina *SOM Retinotopic*

Map (SOMRM). La teoría del modelo LISSOM utiliza la SOMRM para explicar como un mapa cortical genera cierta selectividad a la posición de un estímulo visual en la retina, de forma que el modelo mapea un patrón de entrada en un espacio cortical. La arquitectura de la red se muestra en la Figura 2.20, donde se observan dos capas; La retina y V1. La retina se trata de un conjunto de receptores que contienen el estímulo visual de entrada y V1 es un conjunto de neuronas con pesos sinápticos con valores de [0,1]. La SOMRM funciona como un mapa topográfico que representa el estímulo presentado en la retina localizado en una posición *k*.

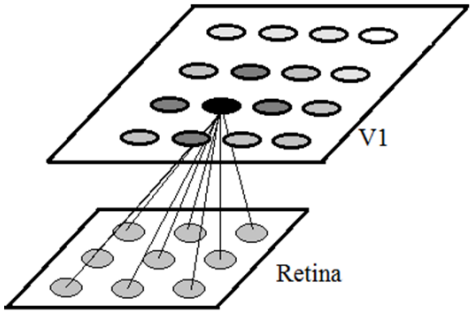


Figura 2.20. Arquitectura de la SOMRM

La arquitectura de esta red es un nivel de abstracción común para representar el funcionamiento de V1 para realizar mapas retinótopicos **[**34**,**77**,**78]. El término retinotópico se refiere a la forma en que se distribuye espacialmente la información que se forma en la retina. Esta estructura retinotópica es la manera en cómo se organizan las áreas bajas de la corteza visual (V1, V2), como en la Figura 2.21a, donde se muestra un mapeo experimental de la retinotopía de la porción del cerebro equivalente a la corteza visual, o en la Figura 2.21b la cual muestra un diagrama de la retinotopía del mapa visual en la corteza estriada del simio, En la Figura 2.21c se aprecia un diagrama de la retinotopía en un cerebro humano y en la Figura 2.21d se observan los dos mapas retinotópicos, el del humano que se encuentra en la parte trasera del cerebro y el del simio, el cual ocupa un área visual separada por un área paracortical[8].

[8] Tejido cortical rodeado por medula.

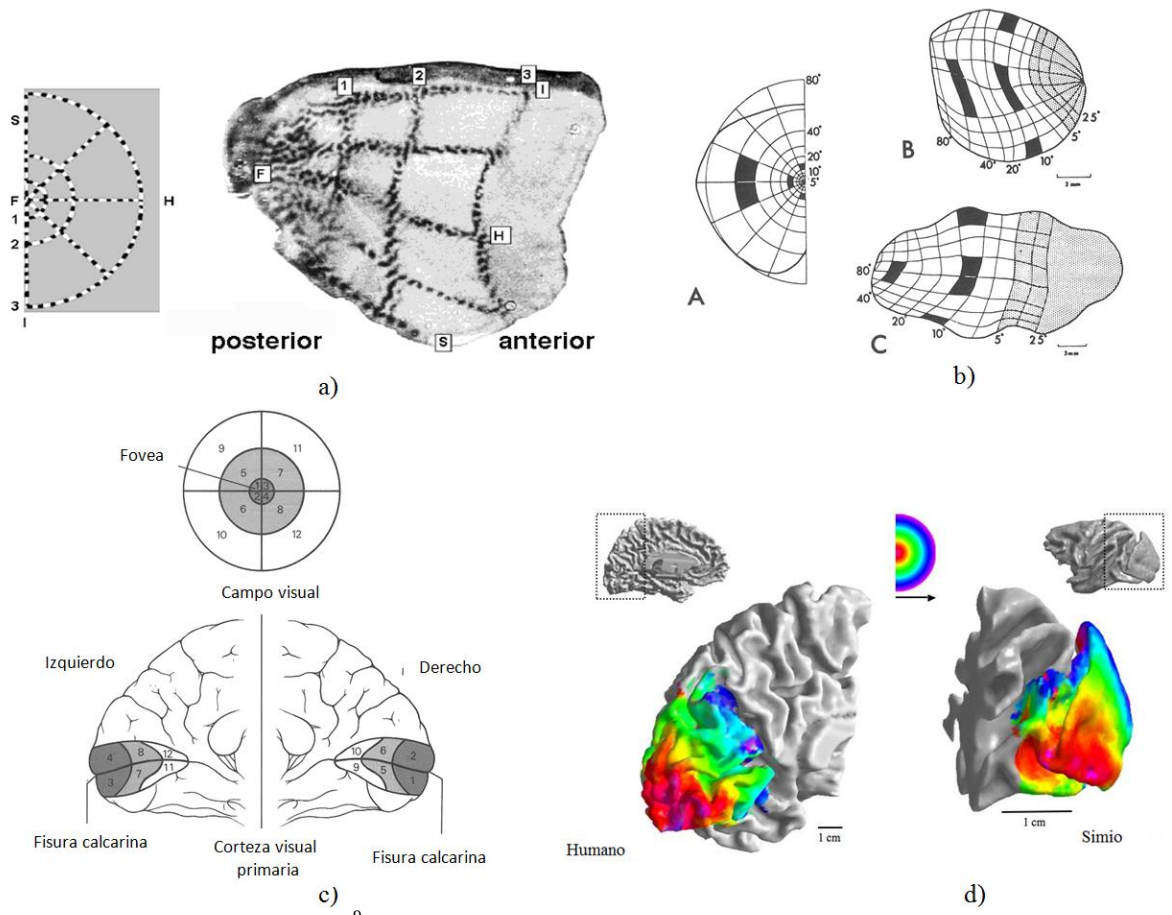


Figura 2.21. Mapas retinotópicos[9]. a). Mapeo experimental por Tootel en 1982. b). Mapeo en el cerebro del simio, donde A es la entrada, B son los LGN y C es la corteza estriada. c). Mapeo en humanos. d). Comparación de dos mapas retinotópicos, donde el área visual del simio está separada del neocortex.

Como se puede observar, la distribución retinotópica está asociada a la estructura espacial de la corteza. En el caso de la SOMRM la distribución también es espacial, como en la Figura 2.20, donde un punto de la retina es mapeado a una entrada de cada neurona. De esta forma, si se tienen *K* receptores o pixeles en la retina, se tendrán *K* pesos en cada neurona. En SOMRM, cada neurona es representada por un vector de pesos $W(i,j)=\{\omega_{1ij}, \omega_{2ij}, \ldots, \omega_{Kij}\}$ donde $W(i,j)$ representa una neurona, cuya respuesta inicial $\eta(i,j)$ en la red es una suma de productos entre la entrada $I(k)$ y el vector de pesos $W(i,j)$; de esta forma la respuesta inicial está dada por:

$$\eta(i,j)=\sum_k I(k)\omega(k,i,j)=IW(i,j) \tag{2.23}$$

[9] Imágenes obtenidas de http://fourier.eng.hmc.edu/e180/lectures/v1/node3.html y www.cns.nyu.edu

donde, (*i,j*) son los índices de neuronas, para la neurona ganadora se utilizará el índice *(i_w,j_w)*, y la respuesta que tenga la mayor magnitud en *$\eta(i,j)$*, es considerada la ganadora, de forma que la neurona ganadora es *$W(i_w,j_w)$*. En este modelo, los pesos se actualizan con la regla dada en [34], la cual es una versión normalizada de la regla Hebbiana dada por:

$$\omega(k,i,j)^{n_s+1} = \frac{\omega(k,i,j)^{n_s} + \alpha I(k)\beta(i,j)}{\sqrt{\sum_k \left(\omega(k,i,j)^{n_s} + \alpha I(k)\beta(i,j)\right)^2}} \tag{2.24}$$

donde α es la taza de aprendizaje (*learning rate*), la cual está dada por:

$$\alpha = 0.42\exp\left(-\frac{6n_s}{T_f}\right) \tag{2.25}$$

T_f es la cantidad de iteraciones que la SOMRM necesita para completar el aprendizaje. n_s son las iteraciones de la SOMRM, β es la función de vecindario que está dada por:

$$\beta(i,j) = \eta(i_w,j_w)\exp\left(-\frac{(i_w-i)^2+(j_w-j)^2}{\sigma_\beta{}^2}\right) \tag{2.26}$$

donde σ_β es el radio de vecindario dado por:

$$\sigma_\beta = \max\left[13.5\exp\left(-\frac{5n_s}{T_f}\right), 0.5\right] \tag{2.27}$$

Implementación de SOMRM en *topographica*. Para entender el funcionamiento de la SOMRM, se recurrió a la plataforma de *topographica*[10]; desarrollado por el Instituto de Neurocomputación Adaptiva de la Universidad de Edimburgo y el Grupo de Investigación de Redes Neuronales de Universidad de Texas de Austin, es un paquete de software para modelar mapas neuronales. El software está diseñado para que diversos investigadores analicen las funciones cerebrales mediante redes auto-organizadas y ofrece una simulación del entrenamiento la SOMRM con una arquitectura dada como en la Figura 2.20. Dicha simulación, tiene como entrada en la retina un patrón gaussiano para entrenamiento de la red, el cual se muestra en la Figura 2.22d. Este patrón va cambiando de posición en cada iteración n_s de SOMRM como se puede observar en las Figuras 2.22a y 2.22d. La red tiene

[10] Disponible en *www.topographica.org*

un conjunto de neuronas *W(i,j)* con pesos aleatorios en el intervalo de [0,1], como se observa en la Figura 2.22c (se aprecian los valores de los pesos en una neurona) y la Figura 2.23a donde se muestran las neuronas de una red de 10x10. En iteraciones iniciales la respuesta de V1 dada por $\eta(i,j)$ genera valores muy similares como se ve en la Figura 2.22b donde se muestra los valores de $\eta(i,j)$ codificados en escalas de grises. Conforme pasan las iteraciones la red va aprendiendo el patrón gaussiano, acumulando en cada neurona *W(i,j)* información de este patrón en las distintas posiciones donde ha estado como se ve en la Figura 2.23b. Sin embargo, en este proceso las neuronas van generando cierta selectividad a las distintas posiciones, de tal manera que en las últimas iteraciones solo la neurona ganadora se actualiza y eso causa que los pesos de cada neurona aprendan el patrón gaussiano pero en distintas posiciones como se ve en la Figura 2.22f y 2.23c donde se observa que en la iteración $n_s=40,000$ SOMRM aprende el patrón gaussiano en distintas posiciones en cada neurona. Este proceso simula la manera en que las neuronas en V1 van estimulando e inhibiendo pesos para generar selectividad a la posición de la información de la retina; es decir, al final de las iteraciones cada neurona en SOMRM va a ser sensible a un espacio específico de la retina.

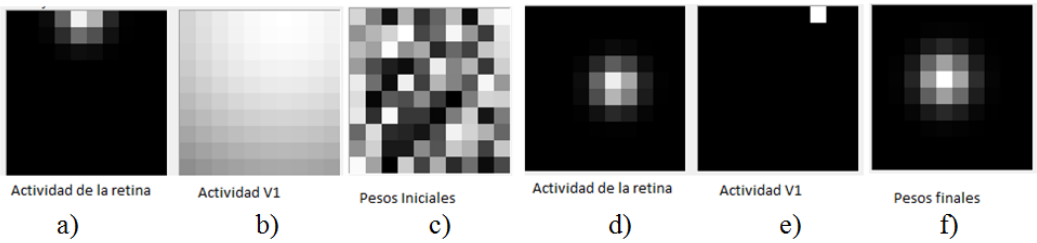
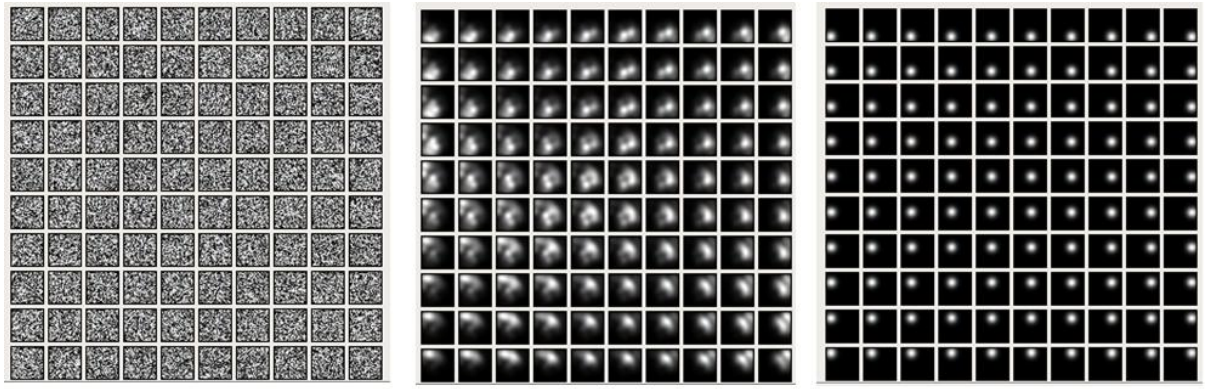


Figura 2.22. Activación de V1 utilizando SOMRM. a). Retina en iteración $n_s=1$. b). Actividad de V1 en la iteración inicial (valores de η). c). Pesos de una neurona en iteración inicial (Pesos *W(i,j)*). d). Retina en iteración $n_s=40000$. e). Actividad de V1 en la iteración final (valores de η). f). Neurona en iteración final (Pesos *W(i,j)*).

Además, la SOMRM implementada en *topographica* estimula un mapa de orientación, el cual es una imagen que describe las orientaciones principales del estímulo de la entrada que proyecta la retina. En la Figura 2.24b se muestra un patrón con forma de media luna proyectado en la retina. Después de 1000 iteraciones, los pesos generan una respuesta similar a la entrada como se ve en la Figura 2.24c. Durante este proceso, un mapa de orientación analizó las orientaciones formadas en los pesos de las neuronas *W(i,j)*. Este mapa se puede observar en la Figura 2.24d; para entender éste código de colores, se debe notar que las orientaciones están dadas por:

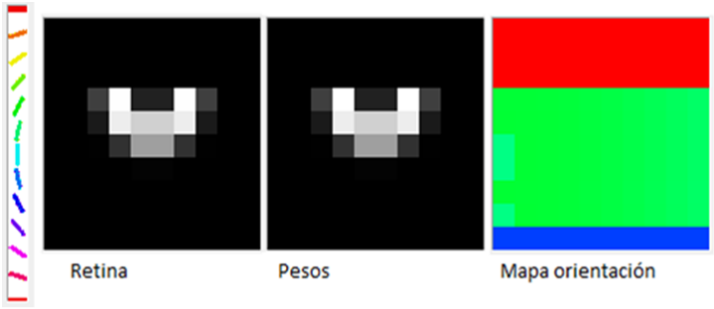
$$Or = Hu/2 \tag{2.28}$$



a) b) c)

Figura 2.23. Red SOMRM. a). Iteración Inicial. b). Iteración Intermedia. c). Iteración final (T_f=*40000)*.

donde *Or* es la orientación y *Hu* es el matiz. De forma tal que el rojo representa una orientación de 0º, el cyan representa una posición vertical de 90º y así sucesivamente con cada matiz que se muestra en el código de colores. Por tanto, el mapa de Orientación de la Figura 2.24d muestra que en la entrada se forma un patrón con orientaciones principalmente horizontal (rojo) y dos orientaciones diagonales (verde y azul) con pendientes de la misma magnitud pero de signo distinto.



a) b) c) d)

Figura 2.24. Mapa de orientación. a) Código de orientación. b). Retina. c). Ejemplo de mapa de pesos después de 10000iteraciones d). Mapa de orientación, donde se observa, se tienen tres orientaciones, horizontal y diagonales.

Análisis del modelo SOMRM. La simulación dada en *topographica* brinda información acerca del comportamiento de SOMRM, pero no da información acerca de la manera en como procesa la información. Por lo tanto, para conocer el funcionamiento de esta red, se realizó una simulación basada en la sección 3.4 de [34], donde se describe el proceso de auto-organización de la SOMRM. En esta sección, el patrón gaussiano que se presenta en la retina está dado por:

$$I(x,y) = \exp\left(-\frac{\left(x - x_c\right)^2 + \left(y - y_c\right)^2}{\sigma_u^2} \right) \tag{2.29}$$

donde σ_u es el ancho de la gaussiana, (x,y) son las coordenadas espaciales que definen la posición de la gaussiana en el receptor *k*. De esta forma, *I(x,y)* es una versión bidimensional del vector I(*k*), donde *k* es:

$$k = y + (x-1)\cdot Y \tag{2.30}$$

donde *Y* es la dimensión máxima de *y*. (x_c,y_c) es el centro de la gaussiana de entrada. Esta gaussiana va cambiando de posición a lo largo de las iteraciones de la SOMRM como lo muestra la Figura 2.25. La retina es representada por una imagen que contiene la gaussiana con un tamaño de 24x24. La capa de V1 (Figura 2.26) es representada por un conjunto de neuronas en un espacio de 40x40, y T_f=40,000. Inicialmente, los pesos tienen valores aleatorios, y conforme n_s aumenta, los vectores de pesos que representan neuronas van formando un conjunto de mapas que van aproximando el patrón de entrada.

Figura 2.25. Patrón de entrada para entrenamiento de la SOMRM documentada en **[**34**]**.

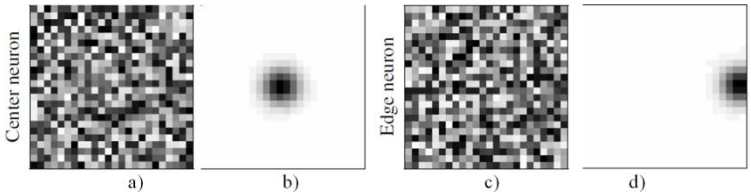


Figura 2.26. Auto-organización de los vectores de pesos en **[**34**]**. a). Pesos iniciales de la neurona que está en el centro de la SOMRM. b). Pesos finales de neurona que está en el centro de (*i*,*j*). c). Pesos iniciales de neurona que está en un extremo en la SOMRM. d). Pesos finales que está en el extremo (*i*,*j*).

Basándose en estos resultados, se procedió a realizar una serie de simulaciones de esta red en MATLAB® para entender el funcionamiento de SOMRM. La entrada *I(x,y)* a la red es un patrón gaussiano dado por la ecuación 2.29. Para procesar dicha entrada por SOMRM, esta se transforma a *I(k)*. En V1, la competición de las neuronas se basa en la ecuación 2.23 y la regla Hebbiana que se utilizó es la que está dada en la ecuación 2.24. El parámetro α es una tasa de aprendizaje que afecta a todas las neuronas (vectores de pesos) con el mismo valor de aprendizaje, además de acuerdo a la ecuación 2.25, tiene un

comportamiento exponencial negativo cuya respuesta se ve en la Figura 2.27a. Esto causa que la regla Hebbiana deje de tener efecto ya que si $n_s \rightarrow \infty$, $\alpha \rightarrow 0$. Como lo muestra la ecuación 2.26, $\beta(i,j)$ caracteriza el aprendizaje en cada neurona y está en función de la distancia de cada neurona $W(i,j)$ a la neurona ganadora $W(i_w,j_w)$ y del radio de vecindario σ_β; donde σ_β tiene un comportamiento exponencial (ecuación 2.26) como se ve en la Figura 2.27b. Si se analiza la ecuación 2.26, $\beta(i_w,j_w)=\eta(i_w,j_w)$ para la neurona ganadora, mientras que en el resto de las neuronas $\beta(i_w,j_w)<\eta(i_w,j_w)$, y conforme $|i-i_w| \rightarrow \infty$ y $|j-j_w| \rightarrow \infty$, el valor de $\beta(i,j)$ en las neuronas no ganadoras tiende a cero. Si se analiza la ecuación 2.27, si $n_s \rightarrow \infty$, σ_β se vuelve constante ($\sigma_\beta=0.5$) y $\beta(i,j)$ dependerá solo de (i,j) y (i_w,j_w). Si $\sigma_\beta=0.5$, para las neuronas no ganadoras $\beta(i,j) \approx 0$ y para la neurona ganadora $\beta(i_w,j_w)=\eta(i_w,j_w)$. Por lo tanto, cuando $n_s \rightarrow \infty$, $\alpha I(k)\beta(i,j) \rightarrow 0$; el aprendizaje de la regla Hebbiana tenderá a ser cero porque $\alpha \rightarrow \infty$ y solo la neurona ganadora podrá actualizarse.

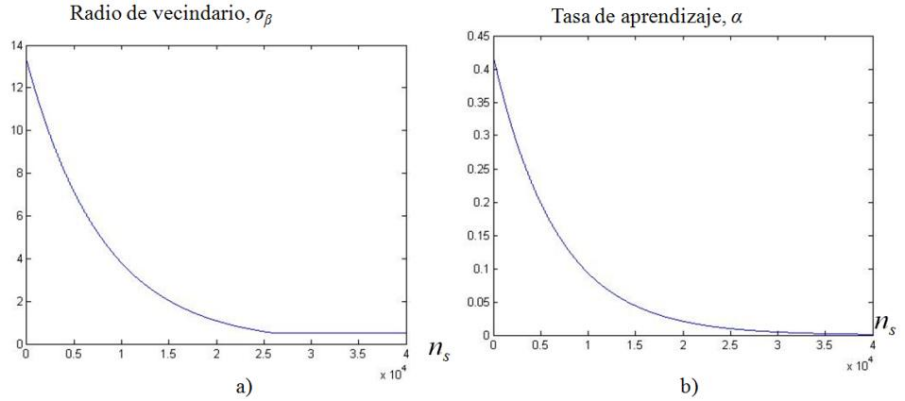


Figura 2.27. Comportamiento de parámetros de la SOMRM. a). Radio de vecindario hasta $n_s=T_f$. b). Tasa de aprendizaje hasta $n_s=T_f$.

Lo descrito anteriormente muestra el comportamiento de los parámetros α y $\beta(i,j)$, pero no explica como SOMRM logra la selectividad. Para este punto, se documentan tres experimentos donde una imagen $I(x,y)$ de 5x5 contiene un impulso $I(x,y)=\delta(x-x_c,y-y_c)$ como se puede ver en la Figura 2.28a. En el primer experimento, se tiene un SOMRM de una neurona (1x1) con pesos inicialmente aleatorios con valores en el intervalo de [0,1]. Esta neurona siempre será la ganadora y $\beta(i,j)$ asignará el mismo valor de aprendizaje a todos los pesos, de manera que, en cada iteración la red va aprendiendo la imagen dada por la ecuación 2.29 en todas las posiciones en las que se encuentra. Por lo tanto, conforme $n_s \rightarrow \infty$, el mapa de pesos va formando un patrón que representa un promedio de $\delta(x-x_c,y-y_c)$ en todas sus posiciones (x_c,y_c) como se muestra en la Figura 2.28b.

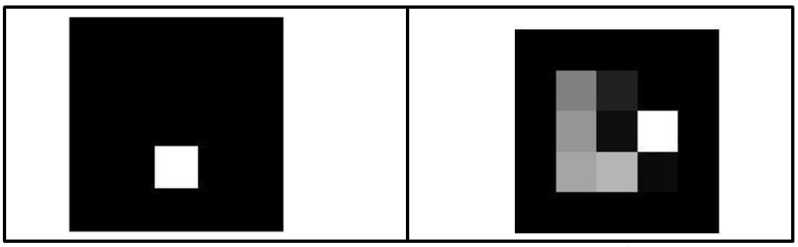


Figura 2.28. SOMRM de 1x1. a). Entrada proyectada en la retina. b). Respuesta de la neurona $W(1,1)$ en $n_s=T_f$.

En el segundo experimento, se utilizó una SOMRM de cuatro neuronas (2x2) y el mismo patrón con la retina de 5x5 del experimento anterior. En este caso las neuronas compiten entre sí; en las primeras iteraciones los valores en $\beta(i,j)$ serán muy parecidos, de manera que todas las neuronas se actualizarán, pero conforme SOMRM itera, solo la neurona ganadora $W(i_w,j_w)$ se actualizará en cada n_s. Ahora bien, la posición de $\delta(x-x_c,y-y_c)$ cambia porque los valores de (x_c,y_c) son diferentes entre una iteración y otra. Por lo tanto, es probable que la neurona ganadora en una iteración, no sea la ganadora en la siguiente. Esto sucede por lo siguiente: si $n_s=1$, los pesos $\omega(k,i,j)$ de todas las neuronas $W(i,j)$ son aleatorios, por lo tanto los valores en $\eta(i,j)$ son aleatorios y la neurona ganadora puede ser cualquiera. Todas las neuronas se actualizan y aprenden parcialmente $\delta(x-x_{c1},y-y_{c1})$ con centro (x_{c1},y_{c1}), aunque la neurona ganadora $W(i_{w1},j_{w1})$ en $n_s=1$ tendrá más parecido al impulso $\delta(x-x_{c1},y-y_{c1})$ que el resto de las neuronas. En $n_s=2$, $\delta(x-x_{c2},y-y_{c2})$ tendrá ahora un centro en (x_{c2},y_{c2}), y en este caso los pesos no serán aleatorios, (tendrán información parcial de $\delta(x-x_{c1},y-y_{c1})$). $W(i_{w1},j_{w1})$ y sus vecinos más cercanos no serán ganadores en $n_s=2$, por dos razones:

1). La regla Hebbiana dada por la ecuación 2.24 causó que $W(i_{w1},j_{w1})$ y sus vecinos más cercanos tengan similitud con $I(x,y)=\delta(x-x_{c1},y-y_{c1})$.

2). Como se puede observar en la ecuación 2.24, el valor de $\eta(i,j)$ se obtiene a partir del producto punto entre cada neurona $W(i,j)$ y la entrada $I(x,y)$. Dado que $W(i_{w1},j_{w1})$ y sus vecinas disminuyeron el valor todos sus pesos excepto en $k_1=y_{c1}+(x_{c1}-1)Y$ a causa de la aplicación de la regla Hebbiana en $n_s=1$, el producto punto va a generar un valor muy bajo en $\eta(i,j)$ para estas neuronas, ya que no coincide k_1 con $k_2=y_{c2}+x_{c2}(Y-1)$. Mientras que la neurona $W(i_{w2},j_{w2})$ más alejada de $W(i_{w1},j_{w1})$ será la ganadora ya que es la que sufrió menos inhibición por la regla Hebbiana en $n_s=1$, por lo que en $n_s=2$, será la ganadora.

En $n_s=3$ y $n_s=4$, las neuronas $W(i_{w1},j_{w1})$ y $W(i_{w2},j_{w2})$ podrán volver a ser las ganadoras si (x_c,y_c) es similar a (x_{c1},y_{c1}) o (x_{c2},y_{c2}). Si no, entonces alguna del resto de las neuronas será la ganadora. Este mismo proceso se repite a lo largo de todas las iteraciones, de manera que las neuronas se van volviendo selectivas a ciertas posiciones. En la Figura 2.29 se observa que después de 1000 iteraciones las neuronas de SOMRM se vuelven selectivas a aprender ciertas posiciones de (x_c,y_c).

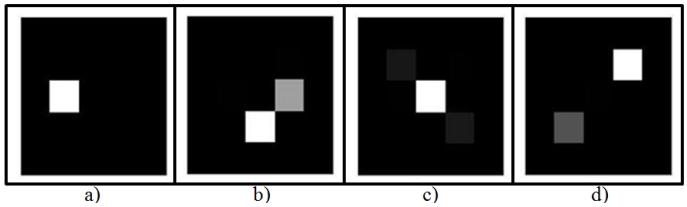


Figura 2.29. SOMRM de 2x2 en $n_s=1000$. a). $W(1,1)$. b). $W(1,2)$. c). $W(2,1)$. d). $W(2,2)$.

Sin embargo, en esta redes de 2x2 σ_β es muy grande para el tamaño de la red y en consecuencia $\beta(i,j)=[0.5,1]$ en todas las neuronas. Esto causa que en las primeras iteraciones los valores de $\eta(i,j)$ sean muy similares entre sí, lo cual hace que sea más aleatoria la selección de la neurona ganadora en las primeras iteraciones. Sin embargo, si se realiza un tercer experimento con una SOMRM de 3x3 utilizando el mismo valor en σ_β, entonces los valores de $\beta(i,j)$ estarán entre [0.05,1], donde $\beta(i,j)=1$ para $W(i_w,j_w)$ y 0.05 para la neurona más alejada de $W(i_w,j_w)$. Por tanto, las neuronas tienen más selectividad ya que en las primeras iteraciones existen neuronas no ganadoras que no inhiben sus pesos, debido a un mejor contraste en los valores de $\eta(i,j)$. Como se observa en la Figura 2.30, después de varias iteraciones las neuronas se vuelven selectivas a posiciones más específicas que el caso de la SOMRM de 2x2.

De esta manera, si $n_s\rightarrow\infty$, cada neurona $W(i,j)$ aprenderá el patrón dado en la entrada pero en una posición distinta. Además de estos experimentos, en esta tesis se realizó una simulación utilizando los parámetros en [34]. En este caso, la imagen de entrada o retina $I(x,y)$ tiene un tamaño de 24x24 y se proyecta un patrón gaussiano dado por la ecuación 2.30 con $\sigma_u=1.5$, y (x_c,y_c) va cambiando de posición en cada iteración como se muestra en la Figura 2.31. La red es de 9x9 neuronas $W(i,j)$ y cada una tiene un conjunto de vectores de pesos retinotópicos con valores en el intervalo [0,1] e inicialmente aleatorios. Por tanto, se tienen 81 vectores de pesos cada uno de 576 elementos. Tal como se mencionó anteriormente, si $n_s\rightarrow\infty$, $\sigma_\beta=0.5$ y $\alpha I(k)\beta(i,j)\rightarrow 0$, y las neuronas se aprenden el patrón

gaussiano de entrada en distintas posiciones tal como se muestra en la Figura 2.32b y Figura 2.32c. Este resultado es muy similar al reportado en [34], como se puede apreciar en las Figuras 2.25 y 2.26, además es similar a la simulación en *topographica*, como se muestra en la Figura 2.22.

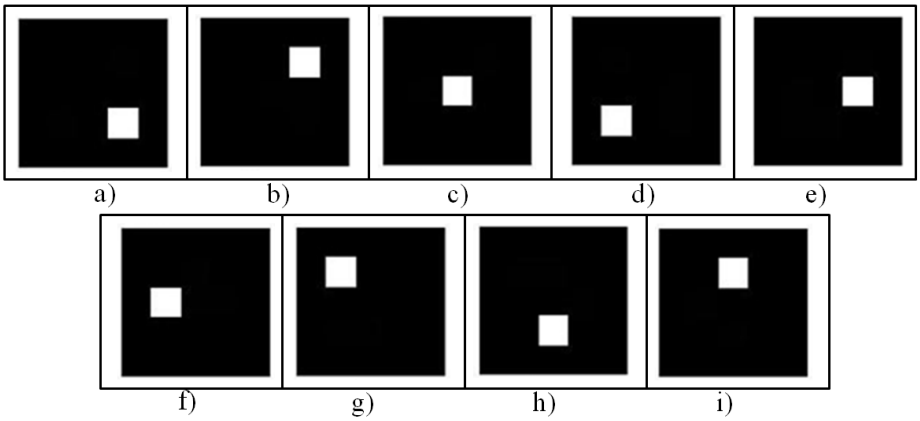


Figura 2.30. SOMRM con nueve neuronas. a). *W(1,1)*. b). *W(1,2)*. c). *W(1,3)*. d). *W(2,1)*. e). *W(2,2)*. f). *W(2,3)*. g). *W(3,1)*. h). *W(3,2)*. i). *W(3,3)*.

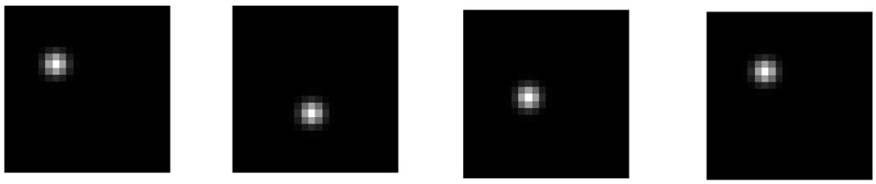


Figura 2.31. Actividad de la gaussiana de entrada a través de las iteraciones de la RESOM.

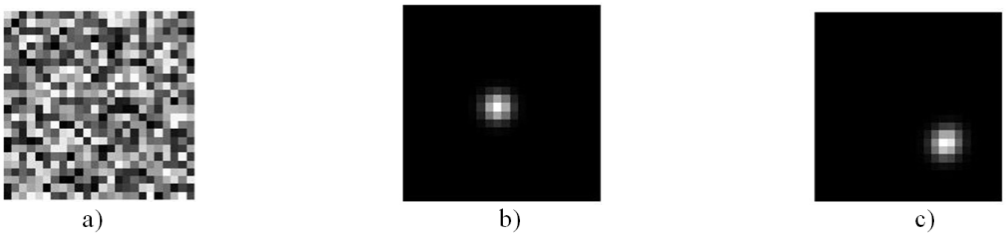


Figura 2.32. Respuesta de SOMRM con 9x9 neuronas. a). Neurona del centro en $n_s=0$. b). Neurona del centro en $n_s=T_f$. c). Neurona del borde en $n_s=T_f$.

La selectividad a la posición de las neuronas tiene el mismo principio que en los experimentos anteriores; en las primeras iteraciones, la regla Hebbiana modifica los valores de los pesos en las neuronas ganadoras de manera que estas se vuelven similares a la entrada *I(x,y)*. Dado que *η(i,j)* es un producto punto y la ecuación 2.29 define en *I(x,y)* valores distintos de cero donde se presenta la gaussiana y cero en el resto de *I(x,y)*, una neurona *W(i,j)* que aprendió parcialmente el patrón gaussiano de entrada con centro (x_{c1},y_{c1}) volverá a ser la ganadora en otra iteración si y solo si el centro (x_c,y_c) es muy similar o igual

a (x_{c1},y_{c1}). Por tanto, al pasar las iteraciones las neuronas van desarrollando una propiedad de selectividad con este patrón de entrada, de manera que al llegar la iteración T_f y α caiga a cero, las neuronas aprenden el patrón gaussiano de entrada pero en las distintas posiciones en las que estuvo a lo largo de las iteraciones. En la Figura 2.33 se puede observar este proceso; en las iteraciones iniciales los pesos de cada neurona son aleatorios, causando que en la respuesta de V1 (valores de $\eta(i,j)$) tenga valores aleatorios (Figura 2.33a). No obstante, al pasar las iteraciones, V1 empieza a generar cierta selectividad a la posición como se puede ver en la Figura 2.33b, donde la respuesta de $\eta(i_w,j_w)$ de $W(i_w,j_w)$ está codificada en blanco y se puede observar en negro aquellas neuronas que no fueron actualizadas. Al final, en la respuesta de V1 solo una neurona y sus vecinas más cercanos se activan para aprender, el resto se inhiben como se observa en la Figura 2.33c. Es decir, conforme $n_s \rightarrow \infty$ el número de neuronas que se actualiza disminuye como en la Figura 2.33b a 2.33d. Dicho estrechamiento continuará hasta que $\alpha \approx 0$ o solo la neurona ganadora se active.

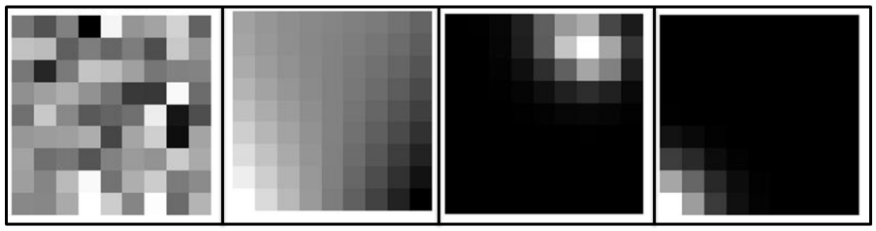


Figura 2.33. Proceso de auto-organización en términos de $\eta(i,j)$. a). $\eta(i,j)$ de cada neurona en $n_s=5$. b). $\eta(i,j)$ de cada neurona en $n_s=100$. c). $\eta(i,j)$ de cada neurona en $n_s=20000$. d). $\eta(i,j)$ de cada neurona en $n_s=40{,}000$.

De esta forma, el resultado de la red de retina 24x24 con 9x9 neuronas se muestra en la Figura 2.34, donde cada neurona aprende la gaussiana de entrada en una posición distinta, similar a la Figura 2.23c. Por lo tanto, se puede concluir que SOMRM es una red basada en el concepto de auto-organización que va aprendiendo un patrón de entrada en ciertas posiciones. Dado que el patrón de entrada como el impulso $\delta(x-x_c,y-y_c)$ o la gaussiana dada por la ecuación 2.30 va cambiando de posición en cada iteración, este proceso de entrenamiento se puede realizar utilizando una secuencia dinámica de imágenes. En consecuencia, el modelo se diseñó para analizar secuencias sucesivas de imágenes, lo cual se puede conceptualizar como una secuencia de video. Por tanto, inherentemente SOMRM es también un modelo plausible para análisis de secuencias de video; aún cuando esta red es un modelo neurocomputacional que se utiliza para ilustrar la selectividad de las

capas corticales bajas en V1, este puede también utilizarse en aplicaciones para análisis de video.

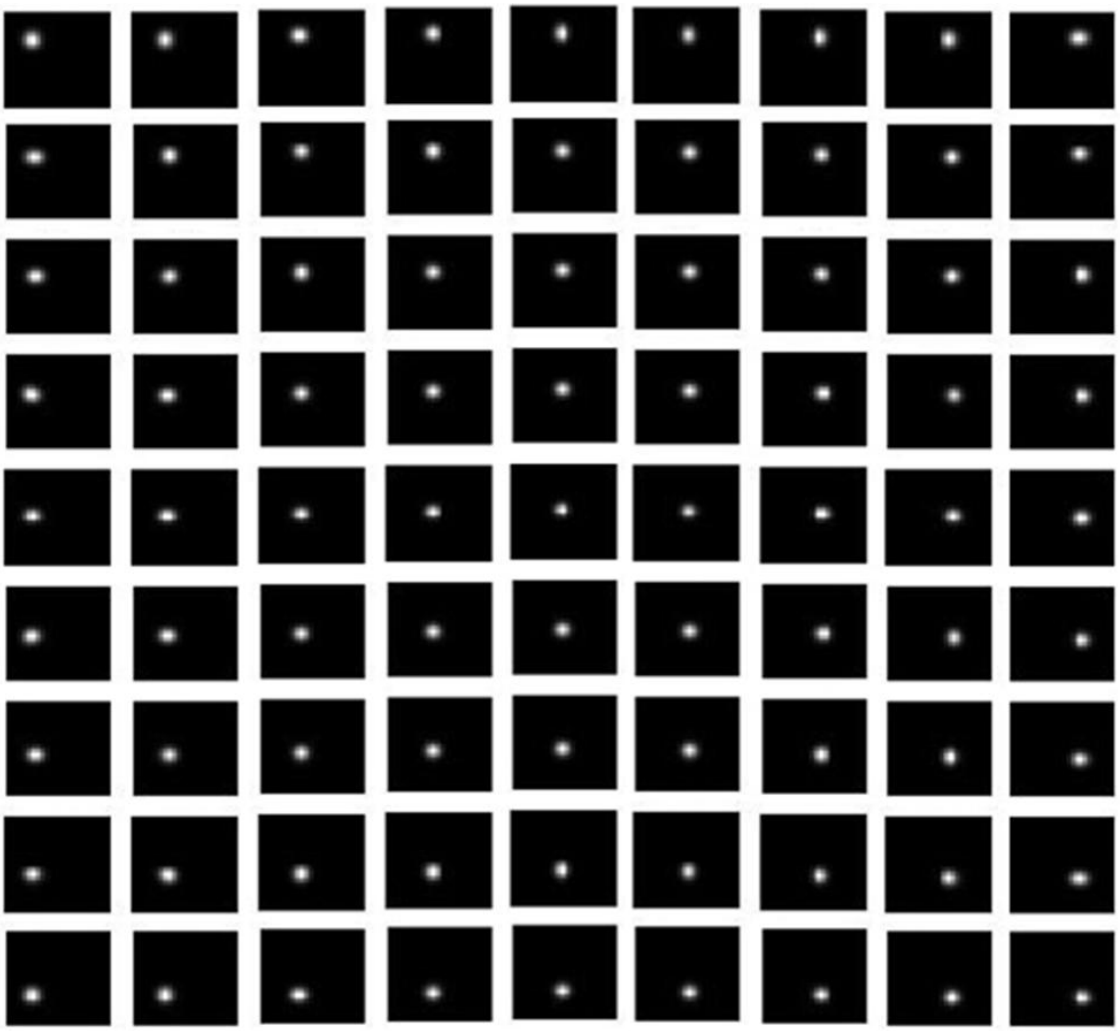


Figura 2.34. Respuesta del modelo SOMRM utilizando como entrada el patrón de la ecuación 2.29, el cual cambia de posición como en la figura 2.33. La retina es de 24x24 y la red tiene 9x9 neuronas, por lo cual 81 mapas de pesos retinotópicos.

2.4.2 Auto-organizado Sinérgicamente y Lateralmente Conectado (LISSOM)

Aunque el modelo de la SOM tiene bases neuroinspiradas y el modelo de la SOMRM se basa en explicar ciertos mecanismos en el entrenamiento de pesos en la corteza visual, ninguno de estos tiene como objetivo describir las partes que conforman la corteza visual.

No obstante, Bednar, Choe, Miikkulainen, y Sirosh desarrollaron un conjunto de modelos para describir ciertas áreas de la corteza visual, el cual llamaron Auto-organizado Sinérgicamente y Lateralmente Conectado (LISSOM) [34]. Estos son modelos auto-

organizados inspirados a partir de varias teorías que sustentan el funcionamiento de la corteza visual como un sistema de organización de pesos basándose en los estímulos de entrada. LISSOM nace como una extensión de la SOM que describe la estructura y el comportamiento de ciertas áreas como la retina, los LGN, la corteza visual y conexiones aferentes (transporta la señal de entrada al sistema nervioso). En LISSOM se asume que la estructura de las cortezas sensoriales se basa en auto-organización de pesos en las conexiones aferentes. Esta auto-organización genera un mapeo de la señal de entrada y genera conexiones laterales que regulan la actividad neuronal. En la Figura 2.35 se observa un diagrama general del modelo LISSOM, el cual inicia su funcionamiento a partir de la retina, luego se pasa a los LGN con los canales ON y OFF y a una representación cortical de *NxN* de V1.

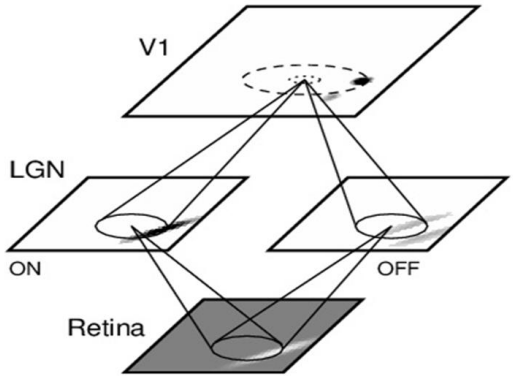


Figura 2.35. Modelo básico LISSOM **[**34**]**.

Una vez que se tiene una entrada por medio de la retina, ésta se procesa por los pesos *L(u,v,a,b)* de LGN ON los cuales están dados por la diferencia de dos gaussianas en cada receptor (*x*,*y*), con centro (x_c,y_c) en el campo receptivo (RF) (*a*,*b*) del canal ON que está dado por:

$$L(x,y,a,b)=\frac{\exp\left(-\frac{(x-x_c)^2+(y-y_c)^2}{\sigma_c^2}\right)}{\sum_{uv}\exp\left(-\frac{(u-x_c)^2+(v-y_c)^2}{\sigma_c^2}\right)}-\frac{\exp\left(-\frac{(x-x_c)^2+(y-y_c)^2}{\sigma_s^2}\right)}{\sum_{uv}\exp\left(-\frac{(u-x_c)^2+(v-y_c)^2}{\sigma_s^2}\right)} \qquad (2.31)$$

La ecuación 2.31 define dos gaussianas, la del centro y la del alrededor, donde σ_c determina el ancho de la gaussiana del centro y σ_s determina el ancho de la gaussiana de alrededor y *u*,*v* es el índice de la respuesta del campo receptivo. Los pesos de la parte OFF

son el negativo de ON. La activación de los LGNs, $\varsigma(a,b)$, se da mediante la suma de la actividad de sus RF, la cual se calcula por:

$$\varsigma(a,b) = \psi\left(\gamma_L \sum_x \sum_y I(x,y) L(x,y,a,b)\right) \tag{2.32}$$

$I(x,y)$ es la activación de la célula (x,y) en el RF (a,b) y γ_L es una constante de escalamiento. La función $\psi(.)$ está definida como:

$$\psi = \begin{cases} 0 & s \le \theta_1, \\ (s-\theta_1)/(\theta_u - \theta_l) & \theta_1 < s < \theta_u, \\ 1 & s \ge \theta_u, \end{cases} \tag{2.33}$$

θ_l y θ_u son umbrales inferior y superior para normalizar $\varsigma(a,b)$ en el intervalo de [0,1], s es el agrumento de $\psi(.)$. Luego, mediante conexiones aferentes, los LGN pasan la información a V1 y éste calcula mediante una sumatoria de pesos dada por:

$$s(i,j) = \gamma_A\left(\sum_{ab \in ON} \varsigma(a,b) A_L(a,b,i,j) + \sum_{ab \in OFF} \varsigma(a,b) A_L(a,b,i,j)\right) \tag{2.34}$$

donde $\varsigma(a,b)$ es el LGN de la neurona (i,j) en los canales ON, OFF. En el caso de $A_L(a,b,i,j)$ es el correspondiente peso aferente y γ_A es un factor de escalamiento constante. La estimulación aferente es procesada mediante una función de activación sigmoidal que tiene la siguiente respuesta inicial:

$$\eta(i,j,0) = \psi(s(i,j)) \tag{2.35}$$

Después del cálculo inicial, la interacción lateral genera una excitación e inhibición definidas como:

$$\eta(i,j)^{n_L} = \psi\left(s(i,j) + \gamma_E \sum_{uv} \eta(i,j)^{n_L - 1} E_L(u,v,i,j) - \gamma_I \sum_{uv} \eta(i,j)^{n_L - 1} I_L(u,v,i,j)\right) \tag{2.36}$$

donde n_L es el índice de iteraciones de LISSOM, $\eta(u,v)^{n_L}$, es la actividad de otra neurona cortical (u,v) durante el paso anterior, mientras que $E_L(u,v,i,j)$ es el peso de excitación lateral hacia la neurona (i,j) e $I_L(u,v,i,j)$ es el peso de conexión inhibitorio. Todas las conexiones de los pesos deben de tener valores positivos. Los factores de escalamiento γ_E, y γ_I representan la fuerza relativa sináptica de las interacciones laterales excitatorias e inhibitorias que normalmente se coloca en un rango de [-0.9,0.9] **[**34**,**79**].** Después de que la

actividad de la red se resuelve, las conexiones aferentes y laterales con los pesos de las neuronas de V1 se modifican de acuerdo a la regla Hebbiana:

$$W(p,q,i,j)^{n_L+1} = \frac{\omega(p,q,i,j)^{n_L} + \alpha X(p,q)\eta(i,j)}{\sum_{uv} \omega(u,v,i,j)^{n_L} + \alpha X(p,q)\eta(i,j)} \qquad (2.37)$$

En este caso, $W(p,q,i,j)^{n_L}$ es el peso actual de la neurona (p,q) a (i,j), $W(p,q,i,j)^{n_L+1}$ es el nuevo peso, α es la tasa de aprendizaje para cada tipo de conexión, $X(p,q)$ es la actividad presináptica. El objetivo básico de LISSOM, es formar una serie de mapas topográficos que se forman a partir de la organización de los pesos entre las neuronas en V1 que describen las principales características de los estímulos visuales [34]. Estos mapas de características son:

Orientación (OR). Conjunto de neuronas que son selectivas a patrones de orientación en bordes y contornos. Éste también se puede combinar con otras características como patrones de dirección y formar mapas de preferencia y selectividad de patrones de orientación.

Color (CR). Conjunto de neuronas que son selectivas a ciertos tipos de colores. Generalmente, este mapa se puede describir en términos de matiz.

Dominancia Ocular (OD). Conjunto de neuronas que determinan la preferencia de cierta área del campo visual por un ojo u otro.

Frecuencia Espacial (SF). Conjunto de neuronas que son selectivas a ciertos tipos de frecuencias espaciales.

Disparidad (DY). Neuronas que describen las diferencias entre las localizaciones de ambos ojos de los objetos.

Implementación de LISSOM en *topographica*. El paquete de *topographica* tiene varias versiones de los modelos LISSOM, entre los cuales, *Gain Control LISSOM* (GCL), que se encuentra en el archivo *gcal.ty*. GCAL es la versión original documentada en **[**34**]** de LISSOM, pero tiene adicionalmente un autoajuste de parámetros en los LGN. La arquitectura del modelo *GCAL* es la misma que la mostrada en la Figura 2.35.

Como se muestra en la Figura 2.36, la retina es un cuadro de entrada que contiene dos gaussianas que cambian de posición de manera espacial en cada iteración. Los LGN

funcionan con pesos fijos de los RF basados en *DoG* dada por la ecuación 2.31 y dan una salida dada por la ecuación 2.32, mientras que V1 auto-organiza los pesos mediante la regla Hebbiana dada por la ecuación 2.37 y la activación de V1 está dada por la ecuación 2.36. En la Figura 2.37 se puede observar que los LGN son sensibles a las polaridades de contraste de manera que el LGN/OFF es sensible a los cambios de obscuridad-luz y los LGN/ON a los cambios luz-obscuridad.

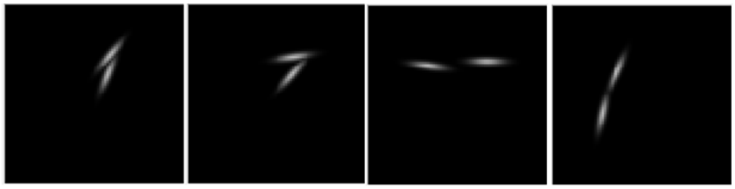


Figura 2.36. Actividad en la retina. a). $n_L=1$. b). $n_L=2$. c). $n_L=3$. d). $n_L=4$.

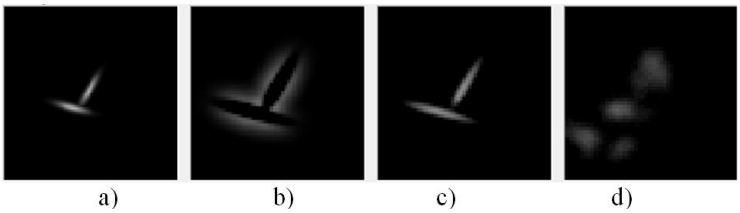


Figura 2.37. Actividad en cada capa del modelo *GCAL*. a). Actividad en la retina en $n_L=0$. b) Actividad en el LGN OFF en $n_L=0$. c) Actividad en el LGN ON en $n_L=0$. d). Actividad en V1.

En la Figura 2.37d, se observa la respuesta de V1, la cual es una combinación entre la respuesta de los LGN, los pesos aferentes A_{LON}/A_{LOFF} (Figuras 2.38a, 2.38b) y laterales E_L, I_L (Figuras 2.38c y 2.38d). Como muestra en la Figura 2.38, todos los pesos son inicialmente aleatorios. Los pesos aferentes A_{LON}/A_{LOFF} se combinan con la respuesta de los LGN como se ve en la ecuación 2.34 para pasar la información a V1. En esta capa, los pesos E_L realizan una estimulación local, los pesos I_L realizan una inhibición global (similar al concepto de LEGION).

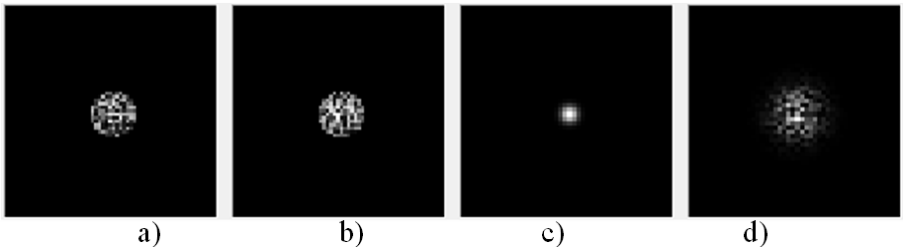


Figura 2.38. Pesos en LISSOM. a). Pesos Aferentes de LGN/ON en $n_L=0$. b). Pesos Aferentes de LGN/OFF en $n_L=0$. c). Pesos estimuladores laterales en $n_L=0$. d). Pesos inhibidores laterales en $n_L=0$.

La salida de la ecuación 2.34 se procesa con una función saturación como se observa en la ecuación 2.35 en la primera iteración. El resultado es un patrón borroso y acuoso como el que se observa en la Figura 2.37d. Los pesos se actualizan con la regla Hebbiana de la ecuación 2.37 y a partir de la segunda iteración, la respuesta de V1 se da por la ecuación 2.36, donde *s(i,j)* se combina con la actividad de V1 y su interacción con los pesos laterales E_L, I_L. Después de 10000 iteraciones, la red se activa como se ve en la Figura 2.39, de tal modo que la actividad en los LGN se mantiene igual, debido a que los pesos que van de la retina a los LGN son fijos. Los pesos de V1 se auto-organizan como se ve en la Figura 2.40, donde se muestran los pesos aferentes que toman un comportamiento similar a los RF ON y OFF que se encuentran entre los LGN y los RGC. En V1 se estimulan una serie de neuronas como se muestra en la Figura 2.39d, y sirven para acotar la información de cada uno de los mapas de características; es decir, los mapas neurales OR, CR, OD, SF y DY van a activan o desactivar neuronas en función de la respuesta en V1.

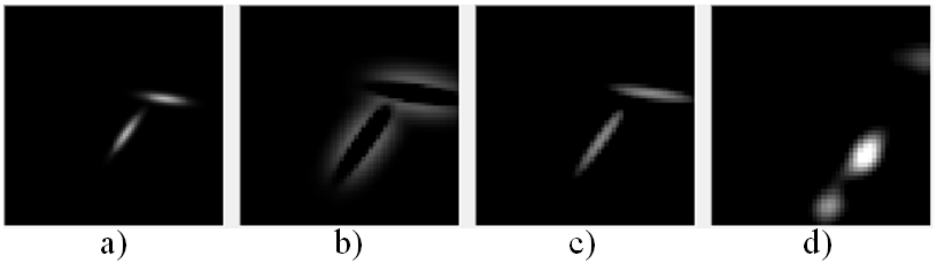


Figura 2.39. Actividad en cada capa del modelo GCAL. a). Actividad en la retina en $t_L=10000$. b) Actividad en el LGN OFF en $n_L=10000$. c) Actividad en el LGN ON en $n_L=10000$. d). Actividad en V1.

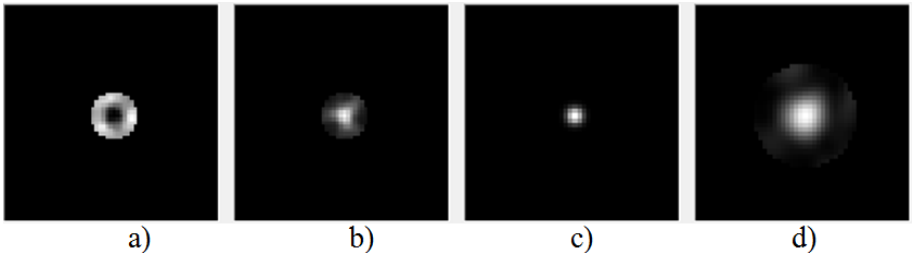


Figura 2.40. Pesos en LISSOM. a). Pesos Aferentes de LGN/ON en $n_L=10000$. b). Pesos Aferentes de LGN/OFF en $n_L=10000$. c). Pesos estimuladores laterales en $n_L=10000$. d). Pesos inhibidores laterales en $n_L=10000$.

En el modelo GCAL, también se analiza el mapa OR junto con otros mapas de preferencia y selectividad (estos mapas se pueden observar en la Figura 2.41). El mapa de orientación es un conjunto de neuronas que responden principalmente a bordes con cierta orientación. Los mapas de preferencia son un conjunto de neuronas sensibles a ciertas direcciones específicas. Los mapas de selectividad muestran direcciones y orientaciones

que son selectivas a los patrones de la retina en una secuencia de tiempo. En *topographica*, se muestran distintos mapas inspirados en el funcionamiento de los mapas de orientación y dirección de la corteza de mamíferos, como los mapas del modelo GCAL (Figura 2.41). Estos mapas se combinan con la respuesta de V1 de forma retinotópica, de manera que V1 puede describir las orientaciones dadas por el patrón de entrada como se observa en la Figura 2.42.

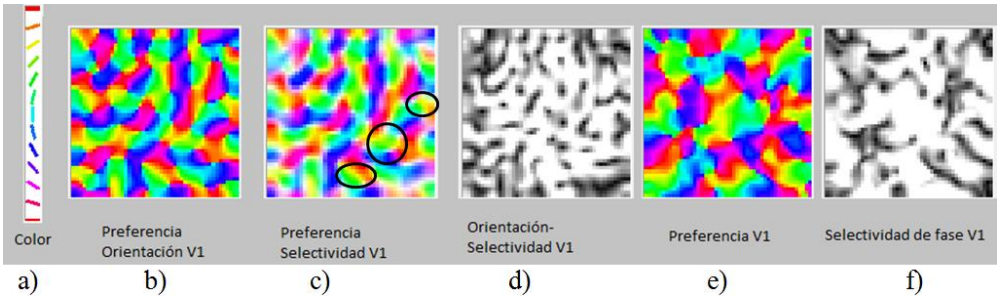


Figura 2.41. Mapa de orientación. b). Preferencia-Orientación V1. c). Preferencia-Selectividad V1. d). Orientación selectividad V1. e). Preferencia V1. f). Selectividad de fase V1.

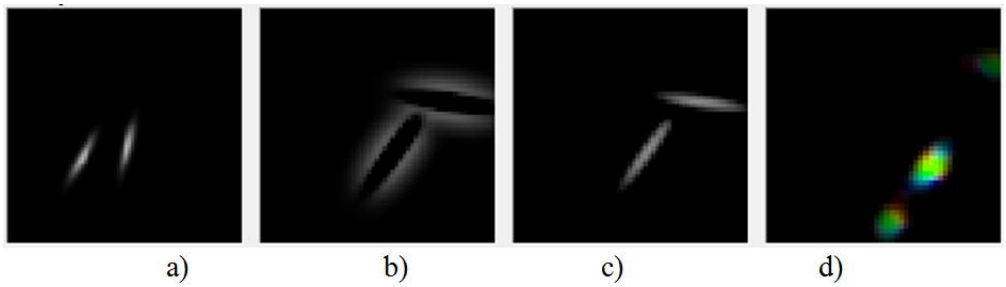


Figura 2.42. Actividad en cada capa del modelo GCAL. a). Actividad en la retina en $t_L=10000$. b) Actividad en el LGN OFF en $t_L=10000$. c) Actividad en el LGN ON en $t_L=10000$. d). Actividad en V1 en $t_L=10000$.

Interpretación de V1 y los mapas de orientación. Cuando V1 se activa, la respuesta $\eta(i,j)^{n_L}$ inhibe o estimula las orientaciones definidas en el mapa de OR, tal como se ve en la Figura 2.42d donde se ven las neuronas que se estimulan y se combinan con el mapa de selectividad de orientación de la siguiente forma:

$$V1\_SPOR = V1 \cdot SPOR \tag{2.38}$$

donde *SPOR* es el mapa de selectividad-preferencia de orientación, *V1_SPOR* es la activación de V1 que muestra la selectividad de orientación ante el estímulo de entrada (Figura 2.42d). En *V1_SPOR* se describe que la orientación de los patrones en la retina es casi vertical (ver ecuación 2.29), es decir, los patrones formados en la retina tienden a presentar valores de pendientes muy altas y positivos, lo cual es evidente en la Figura

2.42a. En la Figura 2.41b se ven encerradas las áreas del mapa de preferencia-selectividad de orientación que se activaron en V1.

Con base a este y otros experimentos, se puede asumir que el modelo LISSOM describe la estructura de la Retina, LGN, V1 así como las conexiones aferentes y laterales entre estas capas. Las capas corticales en LISSOM también se conectan en forma retinotópica como SOMRM. El aprendizaje basado en auto-organización de pesos por la regla Hebbiana modela la estructura de los pesos A_{ON}, A_{OFF}, I_L y E_L. Estos pesos junto con el mapa de OR modela la actividad de V1. Para el modelo GCAL, V1 representa las orientaciones presentes en el patrón visual proyectado en la retina. En otras simulaciones de otros modelos LISSOM, V1 puede representar diversas características del patrón visual proyectado en la retina como orientación, color, disparidad, frecuencia espacial, etc. Por lo tanto, con base al modelo LISSOM se puede concluir que V1 es una capa cortical que mediante auto-organización de pesos que descompone la información del estimulo visual en sus características como orientación, color, disparidad, etc. No obstante, un problema con LISSOM es que en simulaciones realizadas con MATLAB® este utiliza tiempos de procesamiento mayores a un cuadro por segundo.

2.5. Modelos Neurocognitivos

El concepto cognitivo se refiere al procesamiento de la información a partir de la percepción, la atención y el conocimiento adquirido. Los procesos cognitivos permiten generar habilidades complejas como el lenguaje, el aprendizaje, solución de problemas, toma de decisiones y el razonamiento. Este concepto ha sido ampliamente estudiado en áreas como la filosofía y la psicología para entender cómo la mente genera los procesos de razonamiento y aprendizaje. Recientemente áreas como la Inteligencia Computacional y la Inteligencia Artificial han tratado de estudiar este concepto para utilizarlo en el desarrollo de nuevos modelos basados en el comportamiento de los procesos mentales. No obstante, resulta complicado desarrollar esquemas de procesamiento basados en teorías cognitivas, ya que tal como se puede constatar en [39], el desarrollo mental cognitivo es estudiado desde campos muy diferentes, tales como:

- *Filosofía*. Contribuye para explicar la relación de las ciencias cognitivas con el desarrollo y la estructura de la mente, el lenguaje, la lógica y la epistemología.

- *Psicología*. Investiga la forma en cómo se representa la información obtenida a través de los procesos perceptuales y cuáles son los elementos en la toma de decisiones.
- *Neurociencias*. Se encarga de estudiar a nivel del funcionamiento del sistema nervioso los mecanismos para adquirir la información del mundo exterior, como interpreta y guarda dicha información.
- *Inteligencia Computacional*. Estudia el desarrollo de nuevas metodologías para el modelado de esquemas computacionales que se asemejen al comportamiento de la mente. Además, también la Inteligencia Computacional se ha utilizado para simular procesos mentales.
- *Lingüística y lenguaje*. Esta parte es importante en las ciencias cognitivas ya que la mayor parte de las ideas se expresan en función del lenguaje. Además, las ciencias cognitivas estudian cómo se va estructurando el aprendizaje del lenguaje.
- *Cultura y evolución*. Se analiza cuál es el origen de los actuales procesos de razonamiento y aprendizaje, así como las modificaciones de los procesos cognitivos en las diferentes especies animales, y entre seres humanos las distintas sociedades históricas y actuales.

Todos estos campos tratan de analizar los procesos cognitivos bajo diferentes enfoques y a partir de ellos han surgido múltiples teorías sobre el funcionamiento de los procesos mentales que llevan al razonamiento y el aprendizaje. A continuación se analiza un modelo neurocognitivo que cumple con varios de los aspectos mencionados.

2.5.1 Campos de modelado Neuronal (NMF)

Un paradigma reciente propuesto por Leonid Perlovsky es los Campos de Modelado Neuronal (NMF) **[**80**,**81]. En los NMF se consideran diversos aspectos derivados de los procesos mentales tales como percepción, cognición, conceptos, instintos, emociones y pensamiento abstracto. La principal hipótesis de las NMF es que los actuales modelos no están muy lejos de entender el comportamiento de los procesos mentales [82]. Las NMF son sistemas multicapa y heterojerárquicos basados en diferentes tipos de retroalimentaciones entre diferentes capas adyacentes y superiores (por eso el término de heterojerarquica). De acuerdo a [80], las NMF proporcionan una descripción de la percepción biológica describiendo las interconexiones y los mecanismos generados en las

capas corticales, además las NMF incorporan aspectos de los sistemas visuales. En la Figura 2.43 se puede observar el diagrama de las NMF, en el que se pueden ver que se tienen cuatro capas. La primera capa es la entrada de la red y tiene conexiones hacia arriba hasta la primera capa cortical de pesos. Cada neurona n_m tiene un número de sinapsis que reciben señales que se describen de acuerdo a [81] como $X(n_m)=\{X_d(n_m), d=1,...,D\}$, donde D es la dimensión de $X(n_m)$. En este modelo, $X(n_m)$ se describe como un campo de sinapsis neuronales. Los pesos se manejan en términos de dos conexiones; por un lado están las conexiones hacia arriba de los campos de entrada de $X(n_m)$, y por otro lado están las conexiones de regreso de la capa de modelos. Un modelo está descrito como un concepto dado por $M_m(S_m,n_m)$; los modelos se enumeran $m=1,....,M_o$. Cada modelo es caracterizado por sus parámetros S_m, descritos como $S_m=\{S_{am},a=1,...,A\}$. Estos parámetros pueden ser posición, orientación o luminancia de un objeto h. De esta manera, un modelo $M_m(S_m,n_m)$ predice de un valor $X(n_m)$ en una señal de una neurona n_m. Por ejemplo, en el proceso de percepción visual, una neurona n_m recibe una señal $X(n_m)$ de la retina y otra señal de regreso de un primer $M_m(S_m,n_m)$ del concepto de un objeto m. Una neurona n_m se activa si ambas conexiones, tanto de ida como de regreso se estimulan. Esto quiere decir que las señales de regreso de un modelo m deben estar relacionadas de alguna forma con las señales de subida, y este es básicamente una forma simplificada de describir la percepción y la cognición de acuerdo a los NMF. Ambos, percepción y cognición envuelven modelos $M_m(S_m,n_m)$ y aprendizaje, pero en el caso de la percepción, $M_m(S_m,n_m)$ corresponde a objetos, en cognición $M_m(S_m,n_m)$ corresponde a relaciones y situaciones.

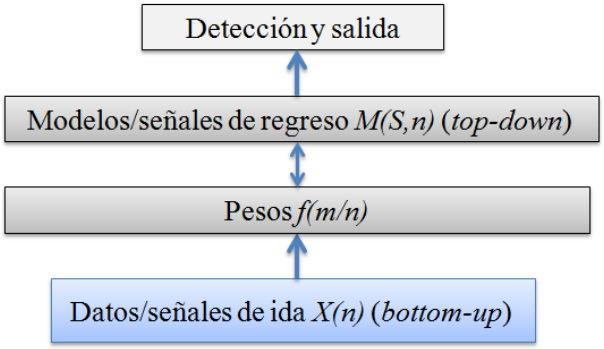


Figura 2.43. Esquema de la NMF.

Respecto al aprendizaje, NMF se maneja en relación a un concepto que se denomina instinto de conocimiento (*KI*), el cual matemáticamente se describe como el incremento de

la similitud entre el conjunto de modelos y los campos de entrada lo cual se puede definir como *L({X},{M})*. Esta medición está dada por:

$$L\left(\{X\},\{M\}\right)=\prod_{n\in N}\sum_{m\in N} r(m)l\left(X(n_m)\middle|m\right) \tag{2.39}$$

donde $l(X(n_m)|m)$ es una medición de similitud condicional entre $X(n_m)$ y el modelo M_m, $r(m)$ es una medición de la representación del objeto *m*. La similitud condicional es una medición de la desviación del modelo con respecto al estímulo de entrada que partiendo del error cuadrático medio, está dado por $[X(n_m)-M(m)]C^{-1}[X(n_m)-M(m)]^T$, donde C^{-1} es la función de covarianza inversa del error. El proceso de aprendizaje de las NMF consiste básicamente en estimar el parámetro *S* y asociar las señales de entrada n_m con el modelo de concepto *m* más indicado maximizando los niveles de similaridad. El mecanismo de entrenamiento da como resultado lo que en **[**81**]** se le denomina Lógica Dinámica (*DL*), el cual consiste en generar un mecanismo "difuso a crisp", que radica en definir las conexiones de regreso como un conjunto difuso y al final del aprendizaje las conexiones de regreso se convierten en conjuntos crisp.

Interpretación de NMF A grandes rasgos el NMF es un sistema neuroinspirado con elementos muy similares a los detallados en otros modelos y métodos difusos, basados en RNA y teoría de probabilidad. No obstante, a diferencia de otros métodos el NMF es una manera de representar aspectos psicológicos y filosóficos del funcionamiento de la mente. A continuación se describirán brevemente algunos de estos.

Mecanismo de conceptos. En esencia, un concepto es el uso de palabras y símbolos para construir una idea, noción u objetos. En NMF los conceptos se describen a partir de los modelos M_m, los cuales terminan de ser construidos adecuadamente cuando las conexiones de regreso se definen como conjuntos rígidos.

Mecanismo de instinto. El instinto es en general el comportamiento inherente de un organismo o unidad biológica. En NMF los instintos se describen como un procesamiento predefinido que no requiere de un proceso de aprendizaje.

Mecanismo de emociones. En NMF esta parte se define como la respuesta o comportamiento del sistema ante una respuesta instintiva. Un aspecto importante de esta respuesta es que afecta directamente en la toma de decisiones de forma tal que la cognición

y la percepción pueden verse afectados. En *NMF KI* se pueden definir ciertos mecanismos similares a las emociones descritas, las cuales están dadas por *dL/dt*, e indica el grado de "satisfacción" e "insatisfacción" del sistema.

Mecanismo de Imaginación. En NMF la imaginación se asume como la excitación de una capa cortical de las cortezas sensoriales en ausencia de una estimulación real. Por ejemplo, la imaginación visual es la estimulación de la corteza visual cuando se tienen los ojos cerrados. En NMF el mecanismo de imaginación es uno de los niveles de procesamiento más alto, ya que en si se trata de la formación de patrones a partir de los modelos M_m.

En general el NMF es un modelado de una RNA basada en teorías de la mente que trata de representar cada uno de los elementos del pensamiento y el razonamiento utilizando teorías derivadas de la Inteligencia Computacional.

2.6. Conclusiones

El objetivo principal de esta tesis es desarrollar un método o un conjunto de métodos de segmentación en secuencias de video que sea compatible con los principios de percepción visual. En esta sección se recapitularán algunos de los principios de percepción analizados en este capítulo. Para ello, los modelos y trabajos descritos con anterioridad serán analizados en términos de las teorías fisiológicas y psicológicas. Luego se presentarán otros modelos neurocognitivos relacionados con la percepción visual.

En la Figura 2.1 se mostró un esquema de cada una de las partes relacionadas con la percepción de movimiento y objetos. Este diagrama parte del análisis en [**32**], donde Yutzumoto ilustra cómo la información visual llega hasta las partes más altas de la corteza visual donde se procesan diferentes tipos de información en diferentes contextos. En la Figura 2.1 se puede observar que V4 tiene conexiones con la corteza temporal (*TempC*), donde se procesa la distribución espacial y detección de objetos, y *TempC* a su vez tiene conexiones con la corteza prefrontal (*PrFC*). Por otro lado MT y MST están asociados a la percepción de movimiento y estas partes de la corteza visual tienen conexiones con la corteza parietal (*ParC*), la cual está asociada al desarrollo de habilidades como análisis espacial, atención visual y toma de decisiones. Además, *ParC* está conectada con *PrFC*, esta última parte de la corteza cerebral correlaciona la información de distribución espacial, detección de objetos, movimiento, memoria y toma de decisiones para desarrollar

habilidades de percepción como detectar objetos peligrosos, detectar la misma letra escrita con diferentes fuentes, detectar objetos desde diferentes perspectivas, etc. Las cortezas *TempC*, *ParC* y *PrFC* tienen muchas otras funciones; por ejemplo *TemC* está asociado con percepción auditiva, el lenguaje y el reconocimiento de rostros, *ParC* tiene funciones relacionadas con la corteza somatosensorial y la coordinación. *PrFC* en cambio es una de las áreas más evolucionadas del cerebro y está asociada también a funciones muy complejas relacionadas con la planificación, la construcción cognitiva, expresión de la personalidad y coordina los pensamientos.

Sin embargo, las cortezas *TempC*, *ParC* y *PrFC* no terminan por si solas el proceso de percepción visual; también tienen conexiones de regreso hasta las células aferentes, tal como lo muestra el esquema de la Figura 2.1. El cerebro genera múltiples habilidades de percepción en detección de objetos mediante un filtrado selectivo de las características de los objetos mediante este proceso de conexiones de regreso, ya que la información dada en *TempC*, *ParC* y *PrFC* modifica el funcionamiento de los niveles más bajos de la corteza visual generando un proceso de atención visual que sirve para filtrar información no necesaria en la percepción de objetos. De modo que, tanto en [32] como en múltiples trabajos neuroinspirados y modelos neurocomputacionales, se asume que este conjunto de conexiones entre diferentes áreas del cerebro juegan un importante papel en los procesos de percepción. A continuación se describirán los mecanismos de las conexiones de regreso de algunos modelos neurocomputacionales.

LAMINART: Como se ha descrito en múltiples ocasiones en esta tesis, este modelo tiene conexiones que retroalimentan cada una de las partes de V2 y V4 para realizar la percepción de objetos. De acuerdo a [41], en V2 existe un conjunto de áreas que mapean los contornos binoculares mediante un filtro de disparidad para generar una proyección de contornos en sus niveles de profundidad. Por otro lado, la formación de los contornos y sus niveles de profundidad tienen conexiones con otras áreas relacionadas con el rellenado de superficies en V2 y V4 para el mapeo de superficies. Uno de los puntos que se sustenta en el modelo LAMINART es la formación del fenómeno de la estereopsis, la cual es una propiedad que tiene el cerebro para la construcción de objetos en superficies tridimensionales. La estereopsis permite a partir de los objetos en V4 construir una escena

espacial (alto, ancho profundidad), así como la construcción de estos objetos para que otras áreas del cerebro puedan calcular distancias y perspectivas entre ellos.

FORMOTION: El primer aspecto que estudia este modelo es el de la formación de patrones de movimientos eliminando el problema de apertura de los campos receptivos. Tal como se documentó en la sección 2.2.2, el modelo FORMOTION proyecta patrones espacio-temporales de todos los movimientos derivados de los cambios de iluminación en V1. Adicional a esto, en V2 se detectan los contornos, formas y niveles de profundidad. Los niveles de MT, integran la información espacio-temporal del movimiento con las formas de los objetos, de manera que se correlaciona la información de los objetos con su movimiento. En MST el modelo FORMOTION inhibe los movimientos pocos significativos y encuentra la dirección de los objetos más significativos de la escena.

LISSOM: El conjunto de modelos LISSOM define diferentes tipos de retroalimentaciones para explicar el funcionamiento de las capas más bajas de la corteza visual. Un modelo que se analizará posteriormente llamado *Tricromatic* LISSOM (TLISSOM) y se utiliza para describir los elementos primarios relacionados con la percepción. Otro modelo relacionado con los principios de percepción es *Hierarquical* LISSOM (HLISSOM) [34]. En la Figura 2.44 se puede observar un diagrama de la arquitectura de este modelo, el cual es muy similar al modelo original LISSOM, pero tiene otras capas corticales como FSA y PGO. HLISSOM se utiliza básicamente para definir los elementos corticales que permiten la detección de rostros en infantes. El módulo FSA contiene un conjunto de neuronas selectivas para la localización de un rostro, mientras que el modulo Punto-Geniculo-Occipital (PGO) es un generador de señales inspirado a partir del funcionamiento de las ondas REM (*Rapid Eyes Movement*) [83]. HLISSOM sugiere que el desarrollo de las habilidades perceptuales en infantes se da principalmente en la etapa REM mientras estos duermen, ya que los pesos se organizan con base a los estímulos de entrada más relevantes presentados durante la vigilia.

Otros modelos: Los modelos descritos tratan situaciones muy diferentes relacionadas con distintos mecanismos de percepción. Tanto en estos modelos como en otros descritos en **[**84**,**85**,**86**,**87**]** **s**e sugiere que muchas de la habilidades cognitivas que definen procesos perceptuales se pueden desarrollar mediante modelos auto-organizados con conexiones

hacia arriba (*bottom-up*) y de regreso (*top-down*). Estas últimas conexiones juegan un papel importante en los procesos perceptuales ya que en los modelos de **[**84**,**85**,**86**,**87**]** las capas corticales tienen un aprendizaje no supervisado, pero las conexiones de regreso funcionan como un factor de retroalimentación que reduce la incertidumbre, la información no relevante [85] y modulan el comportamiento cortical para adaptarse a un estímulo esperado [84]. Además, las conexiones de regreso también tienen pesos que se adaptan para estimular capas corticales más bajas para obtener información de estímulos esperados como se ve en la Figura 2.45. Con base a estos trabajos, se puede observar que las conexiones de regreso pueden influir en el comportamiento de las capas corticales bajas, de forma que los parámetros de las capas corticales bajas pueden ser estimados automáticamente mediante el resultado de capas corticales altas mediante las conexiones de regreso.

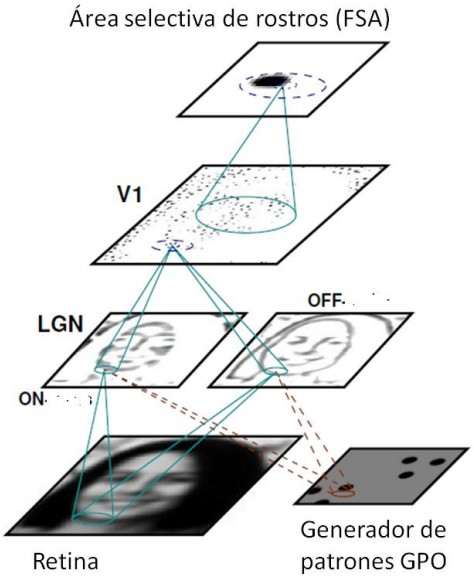


Figura 2.44 Arquitectura del modelo HLISSOM. La parte de GPO genera señales únicamente cuando el modelo HLISSOM "duerme" **[**83**]**.

Modelos Pulsantes y teoría de la Gestalt. Como se vio en la sección 2.3, una de las principales teorías en psicología que sustentan muchos de los principios de la teoría de la *Gestalt*. Existen múltiples trabajos como en [88] donde el área del procesamiento de imágenes se utiliza para estudiar aspectos de la psicología perceptual mediante los principios de la *Gestalt*. Además, en la literatura existen múltiples trabajos como en **[**89**,**90**,**91**,**92**]** en los que los principios de la *Gestalt* se utilizan para sustentar esquemas de diferentes métodos de procesamiento de información visual. Entre los múltiples trabajos que sustentan principios de esta teoría están los modelos basados en RNA pulsantes, ya que

en estos modelos se argumenta que la sincronización de las oscilaciones o pulsos pueden representar los objetos o características de ellos de una escena visual, lo cual se puede ejemplificar con la implementación de la PCNN de la sección 2.3.2. De acuerdo a **[**46**,**47**,**48**,**57**]**, esta sincronización neuronal garantiza el principio de agrupamiento de la *Gestalt* ya que esta sincronización va formando las características de cada uno de los objetos y posteriormente se van generando cada uno de los objetos en un escenario.

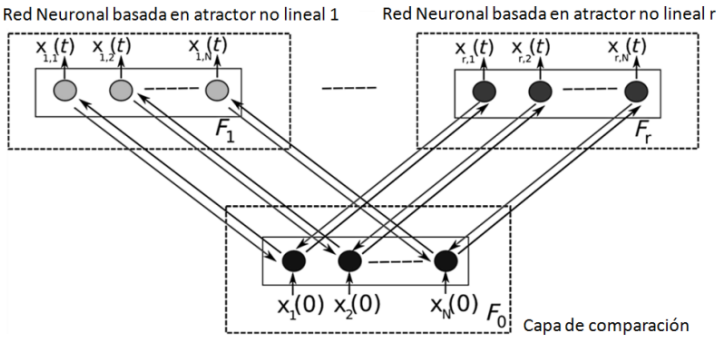


Figura 2.45. Red propuesta en **[**87**]**.

Los modelos descritos en esta sección serán utilizados como base para el desarrollo de modelos y métodos de segmentación. En los siguientes capítulos se propondrá un modelo basado en SOMRM que simula las partes bajas de V1 y puede aplicarse en segmentación de movimiento. Además, basándose en ciertos conceptos de FORMOTION se propondrá un método de segmentación de objetos dinámicos. Las redes pulsantes y algunos modelos perceptuales se utilizarán en el capítulo V para proponer un método de segmentación de objetos estáticos. A partir del capítulo III, el enfoque será un poco distinto ya que este capítulo se enfocó a presentar modelos y redes basados en la corteza visual, mientras que en los siguientes el enfoque será solucionar problemas de segmentación basándose en los modelos presentes en este capítulo.

III. MÉTODO DE SEGMENTACIÓN DE OBJETOS DINÁMICOS

Entre las aplicaciones basadas en computación neuroinspirada, está el análisis para detección de movimiento, siendo común el uso de RNA en ocasiones basadas en los mismos principios de algunos de los modelos neurocomputacionales de la sección anterior. Ejemplos de redes utilizadas para este análisis son PCNN [93], CNN **[**16**,**17**,**94**]**, PNN [95] y SOM **[**13**,**14**,**15**,**96**,**97**]**. Concerniente a trabajos relacionados a estas redes, en [93], se diseñó un método basado en PCNN para detectar movimiento mediante codificación de pulsos de forma similar a lo que realiza la corteza visual. Referente a trabajos relacionados con CNN, esta red es aplicada para percepción de alerta y atención en [94]. Además, CNN se ha utilizado en múltiples trabajos relacionados con segmentación de movimiento como en **[**16**,**17**,**98**].** En [95], se presenta la *Background neural network* (BNN), una variante del modelo tradicional PNN que realiza un análisis estadístico para clasificar el estado de los pixeles si estos contienen información de objetos fijos o en movimiento. Dicha clasificación se utiliza para realizar un modelado de fondo, el cual es una técnica que consiste en modelar un video que contenga los objetos que permanecen fijos en una secuencia de video. El resultado se utiliza para detectar los objetos en movimiento mediante la diferencia entre la secuencia de video y el modelo de fondo. En [13] se presenta la SOBS (*Self-Organizing Backgorund Subtraction*), una red neuronal SOM para modelado de fondos dinámicos en escenarios obtenidos de sistemas de vigilancia. SOM es una red neuronal comúnmente utilizada en el modelado de fondos dinámicos (fondos que cambian a lo largo de la secuencia de video), tal como se puede ver en **[**14**,**15**,**96]. Aunado a esto, SOM es una red que ha sido utilizada para recrear los mecanismos de percepción de movimiento basándose en principios neuroinspirados, como en **[**97**,**76]. Con base a estos trabajos y en la literatura analizada, se puede observar que existen dos formas de realizar análisis de movimiento con RNA y modelos neuroinspirados; sistemas de percepción de movimiento como en **[**76**,**93**,**94**,**97], y sistemas de modelado de fondo para segmentación de objetos en movimiento como en **[**13**,**14**,**15**,**95].

Además de los modelos basados en RNA para simular los mecanismos de percepción de movimiento, existen modelos neurocomputacionales para simular los mecanismos de percepción al nivel de la corteza visual, tal como se vio en la sección 2.2.2, donde se

documenta el modelo FORMOTION; un modelo que trata de explicar la detección de objetos en movimiento mediante dos procesos. El primero es la detección de formas que se basa en encontrar contornos de objetos y mapearlos en superficies de profundidad. El segundo es la detección de cambios de luminancia y direcciones que se basa en encontrar patrones de movimiento. Ambos procesos en FORMOTION se combinan para la detección y seguimiento de objetos en movimiento, formando los patrones de movimiento de cada objeto en la escena.

De acuerdo a la literatura analizada, los trabajos relacionados a detección de movimiento basados en principios neuroinspirados se pueden categorizar en aquellos que se utilizan para detección de movimiento en secuencias de video y en aquellos que se utilizan para simular los mecanismos de percepción de movimiento en la corteza visual. De los trabajos relacionados en detección de movimiento, sobresale CNN para mejorar la detección de objetos dinámicos y la SOM para modelado de fondo. Por tanto, en este capítulo se presenta un método de segmentación de objetos dinámicos denominado SOM-CNN, el cual realiza un modelado de fondo mediante una red propuesta en esta tesis denominada SOM Retinotópica (RESOM) y una binarización para realizar una segmentación dura mediante otro modelo también propuesto llamado Red Neuronal Celular de Umbralización por Vecindario (NTCNN). A diferencia de otros métodos basados en SOM, RESOM está inspirada en los mecanismos de aprendizaje de las capas corticales de células simples en V1; esto debido a que dicha red es una versión del modelo neurocomputacional SOMRM (subsección 2.4.1) para análisis de videos con escenarios reales. NTCNN está inspirada en la arquitectura del modelo perceptual para detección de movimiento de [**94**] y en el análisis de superficies de la ecuación (2.4) del modelo LAMINART. Para esta tesis, segmentación de objetos dinámicos se refiere a encontrar regiones que representen los objetos que puedan cambiar de posición en una secuencia de video.

Este capítulo se divide en seis secciones: La sección 3.1 muestra el escenario experimental y explica la técnica del modelado de fondo para segmentación dinámica. La sección 3.2 presenta SOM Retinotópica (RESOM) una novedosa red neuronal para análisis de video, y el método de modelado de fondo basado en RESOM llamado DR-SOM. La sección 3.3 presenta la RNA Celular de Binarización por Vecindario (NTCNN), una red

neuronal para binarización que define umbrales a cada elemento de entrada con base a la información del vecindario. En la sección 3.4 se presenta el método de segmentación de objetos dinámicos llamado SOM-CNN. En la sección 3.5 se presenta la validación y caracterización realizada a SOM-CNN. Finalmente la sección 3.6 reporta las conclusiones del capítulo.

3.1 Modelado de Fondo para Segmentación de Objetos Dinámicos y escenario experimental

Percepción de movimiento es un tema analizado desde diferentes enfoques. Por ejemplo, de acuerdo **[**94**]** la percepción de movimiento está basada en alerta visual, la cual se trata de un proceso para detectar los patrones dinámicos más sobresalientes en un escenario. El modelo FORMOTION propone que la percepción de movimiento se genera a partir de la retroalimentación de la estereopsis y detección de formas en V2 y la detección de patrones de movimiento en MT a partir de cambios de luminancia **[**42**]**. En **[**99**]** se propone que la percepción visual humana está separada en tres partes: un sistema que responde a los cambios de luminancia, un sistema que responde a cambios de texturas y un sistema que genera un mapa de sobresaliencia. Este último sistema realiza una separación entre las figuras en movimiento y el resto de las figuras en el escenario. Como se puede ver en estos trabajos propuestos, la percepción de movimiento es analizada desde diferentes enfoques. Al igual que estos trabajos, en la literatura existen muchos enfoques para estudiar la percepción de movimiento ya que la percepción de movimiento se da a través de la integración de múltiples tareas como detección de objetos en movimiento, detección de automovimiento, discriminación de movimiento de retina, etc **[**100**]**. No obstante, en **[**42**,**94**,**101] y otros trabajos analizados relacionados a detección de objetos en movimiento, tienen en común ciertos aspectos sobre los cuales se pueden plantear dos hipótesis. La primera de ellas es que el sistema visual genera mapas de sobresaliencia para detectar los objetos en movimiento, lo cual es una idea obtenida de **[**94**,**99**].** Otra hipótesis es que la detección de objetos en movimiento se forma a partir de la interacción entre la información de forma y cambios de luminancia, lo cual es una idea obtenida de [42].

En la literatura analizada, el método de segmentación de objetos dinámicos de substracción de fondo (*background subtraction*) se asemeja a estos principios ya que la

diferencia entre la secuencia original y los objetos de fondo genera un mapa que detecta los objetos en movimiento y este se asemeja al concepto de alerta visual de [94] o de sobresaliencia de [99]. Sin embargo, este método solo se puede utilizar en aplicaciones donde los videos hayan sido adquiridos con cámaras estacionarias ya que es necesario tener un escenario estático que se pueda utilizar para modelar el fondo. Dado que existen múltiples aplicaciones en la literatura y en la industria relacionadas con uso de cámaras estacionarias, SOM-CNN se basará en substracción de fondo. Para dar plausibilidad con modelos perceptuales, SOM-CNN utiliza un análisis de cambios de luminancia para definir el comportamiento de los parámetros. Es decir, con la substracción de fondo SOM-CNN va a poner atención a todos los objetos en movimiento y su forma (generando la alerta visual), mientras que los cambios de luminancia brindarán información del posible origen de los patrones de movimiento. Esto último es para que SOM-CNN pueda discernir si los objetos en movimiento detectados son realmente los objetos dinámicos de interés o no. En la mayor parte de las aplicaciones que utilizan Substracción de fondo, la detección de movimiento se aplica para separar en dos clases los objetos que se encuentran en el escenario, las cuales son:

Objetos estáticos. Son aquellos objetos que pertenecen al fondo ya que permanecen fijos por periodos muy largos en la secuencia de video o siempre están en la misma posición.

Objetos dinámicos. Son aquellos objetos que pueden cambiar de posición durante la secuencia de video. Ejemplos de estos objetos son personas, automóviles, trenes, objetos personales, etc.

En el diseño y evaluación del desempeño de SOM-CNN, es necesario buscar una aplicación donde se pueda utilizar este método. La aplicación seleccionada es monitoreo inteligente de movimiento en secuencias de video, la cual ha sido en los últimos años particularmente importante en áreas como rehabilitación, la industria y sistemas de vigilancia relacionados con seguridad, tráfico automotriz o marítimo. Para realizar este monitoreo inteligente, es común adquirir los videos a partir de cámaras estacionarias, ya que es una forma barata de obtener información suficiente de un escenario de interés y los eventos que trascurren en él a través del tiempo. No obstante, para analizar este tipo de videos es necesaria la intervención humana, por ejemplo un sistema de vigilancia se

compone de múltiples cámaras que proyectan diferentes videos en monitores que son vigilados por operadores, de manera que cada operador tiene que analizar múltiples monitores [101]. Esta intervención humana no solo es poco eficiente, además es costosa [102]. Por tanto, una forma de aplicar los métodos neuroinspirados resultantes de este trabajo de tesis es adaptarlos para solucionar problemas de monitoreo inteligente en secuencias de video en los cuales se pueda minimizar la intervención humana. En la literatura analizada, el monitoreo se utiliza principalmente para la segmentación de objetos dinámicos mediante el algoritmo de la Substracción de fondo para analizar videos de sistemas de vigilancia. En los escenarios obtenidos por estos sistemas, los objetos fijos y en movimiento pueden estar cambiando a lo largo de la secuencia de video, ya que existen varias situaciones que pueden alterar la forma, color y estructura de los diversos objetos del escenario. Estas situaciones se pueden utilizar para poner a prueba el desempeño de SOM-CNN para segmentar objetos dinámicos. Con base a **[**103**,**104**,**105**]**, estas situaciones se categorizan en este trabajo de la siguiente forma:

Cambios de iluminación graduales: El modelado de fondo debe seguir cambios en la escena causados por cambios de iluminación graduales tales como amaneceres, atardeceres, nublados, etc.

Cambios de iluminación repentinos: El modelado de fondo se debe adaptar lo más rápido posible cuando las condiciones de iluminación del escenario cambien repentinamente, como por ejemplo cuando se apagan o encienden luces.

Fondos dinámicos: La substracción de fondo no debe detectar objetos en la escena que son parte del fondo pero exhiben cierto movimiento. Como ejemplos hojas de arboles, fuentes, cortinas, etc.

Camuflaje: Esta situación consiste en la no detección de objetos dinámicos o parte de estos debido a que las características cromáticas entre estos y el fondo son similares. Existen videos con bajo contraste que pueden generar camuflaje en toda la escena, tales como escenarios obscuros o videos adquiridos con cámaras térmicas.

Bootstraping: El modelado de fondo debe aprender solo los objetos estáticos, aún cuando los objetos dinámicos estén al inicio de la secuencia de video.

Ruido de video: Se deben evitar problemas de segmentación causados por baja calidad del video.

Movimiento de cámara: El modelado de fondo debe seguir los cambios causados por vibraciones no deseadas de la cámara para no tener errores significativos en la segmentación.

Objetos dinámicos estacionarios: Deben ser detectados objetos dinámicos que permanecen en la misma posición por un periodo finito de tiempo menor a la duración total de la secuencia de video.

En la literatura existen muchos métodos para desarrollar modelados de fondo que resuelven una o más de las situaciones que se acaban de describir. De los métodos analizados en la literatura, destacan los métodos para resolver problemas de fondos dinámicos y sombras basados en SOM **[**13**,**14**,**15**,**96**]**, Redes Neuronales Probabilísticas **[**95**,**106] y otros métodos basados en modelos probabilísticos como filtros *Kalman* [11], Redes Bayesianas [107] o Algoritmos de Muestras Gaussianas (GMM) **[**10**,**108**]**. Existen otros métodos que se enfocan a resolver problemas de cambios de iluminación repentina como **[**109**,**110**,**111**,**112**].**

Respecto al hardware en sistemas de vigilancia, cualquier cámara inteligente, CCTV o embebida, adquieren video en el espacio de color RGB (*R* es el canal rojo, *G* es el canal verde y *B* es el canal azul). Por tanto, para este trabajo un cuadro en una secuencia de video es $I(x,y,z)^t$, donde *x,y* son las coordenadas de los pixeles en el cuadro, *z* representa los canales de color, y *t* el índice de cada cuadro en la secuencia de video.

Para el diseño de SOM-CNN, fueron seleccionados diversos videos de bases de datos utilizados en la literatura, las cuales fueron *wallflowers*[11], *perception*[12], *LIMU*[13] y *PETS*[14]. En la Figura 3.1 se puede observar el primer cuadro de los videos utilizados, todos ellos con una resolución de 120x160 y128x160. Los videos '*CAM*', '*WS*', '*WT*', '*FT*' 'SS' y '*MR*' fueron seleccionados por tener escenarios con objetos de fondos dinámicos. Los videos '*TD*', '*LB*', '*LS*' y '*CM*' se utilizaron porque tienen cambios de iluminación, además '*TD*' y

[11] Disponible en http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm
[12] Disponible en http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[13] Disponible en http://limu.ait.kyushu-u.ac.jp/dataset/en/
[14] Disponible en http://www.cvg.rdg.ac.uk/slides/pets.html

'*CM*' contienen escenarios propicios para camuflaje ya que en partes del video se vuelven muy obscuros. Los videos '*P2009_L1_1*', '*P2009_L1_5'* y '*P2006_3_C2*' tienen objetos dinámicos al inicio de la secuencia. El resto de los videos se eligieron para probar el método propuesto en diversos escenarios reales exteriores e interiores. Los videos '*PT1B*', '*P2001_1*', '*P2009_L1_1*', '*P2009_L1_5'*, y '*P2006_3_C2*' fueron obtenidos de la base de datos de *PETS*, los videos '*MO*', '*TD*', '*CF*', '*LS*' y '*WT*' fueron obtenidos de la base de datos de *Wallflowers*. Los videos de '*CAM*', '*LB*', '*FT*', '*MR*', '*WS*' y '*SS*' fueron obtenidos de *perception*, los videos de '*CM*' y '*AB*' se obtuvieron de *LIMU*. El resto de los videos fueron obtenidos a partir de otros trabajos.

Figura 3.1. Primer cuadro de los videos utilizados en este capítulo.

Los videos utilizados para las pruebas tienen frecuencias de cuadros (*frame rate*) de 15 o 25 fps. Para la validación de SOM-CNN, se utilizaron las bases de datos de *changedetection*[15], las cual se describirá en la sección 3.5, mientras que los videos seleccionados para diseño de SOM-CNN se utilizarán en las secciones 3.2 a 3.4 para explicar el diseño de la substracción de fondo y el diseño de SOM-CNN.

[15] http://changedetection.net/

3.2 Diseño de SOM Retinotópica (RESOM) y Modelado de Fondo

Básicamente, el método de substracción de fondo se divide en dos etapas: modelado de fondo y diferencia entre el fondo y el video original. Para el modelado de fondo se utiliza una RNA propuesta en esta subsección basada en SOM que se describe a continuación.

Como se mencionó anteriormente, SOM es un modelo utilizado para hacer simulación de partes corticales en la corteza visual y para aplicaciones en visión por computadora y ha demostrado en **[**13**,**14**,**15**,**96**]** ser un método exitoso en modelado de fondo. Por tanto, dado que se busca generar un modelo neuroinspirado, en esta tesis se recurre a SOM para el modelado de fondo, tomando como base SOM *Kohonen* y los modelos LISSOM.

Entre los modelos LISSOM, SOMRM es el que recrea de la forma más simple posible la manera en que V1 realiza mapas topográficos de la retina y ejemplifica el proceso de organización de V1. Si SOMRM utiliza como entrada una secuencia de video con un escenario real, este será aprendido por todas las neuronas, pero en cada neurona habrá selectividad a ciertos patrones de entrada. Basado en SOMRM y sus características de aprendizaje, en esta sección se presenta un método para modelado de fondo que se utiliza para segmentación, donde el criterio de selectividad en las neuronas es aprender solo los objetos que permanecen fijos. El método propuesto denominado SOM Retinotopica Dinámica (DR-SOM) se muestra en la Figura 3.2, donde se puede apreciar que la entrada es un cuadro de una secuencia de video $I(x,y,z)^t$, donde $z=\{R,G,B\}$. Los tres canales RGB son las entradas para una red neuronal propuesta en esta tesis llamada SOM Retinotópica (RESOM), la cual consiste en tres vecindarios de Mapas Auto-organizados Retinotópicos basados en SOMRM, cada uno con cuatro neuronas representadas por los mapas retinotópicos de pesos $w(k,i,j)^z$. Es decir, el canal rojo vectorizado en $I(k,R)^t$ es la entrada para las neuronas $w(k,i,j)^R$, el canal verde vectorizado en $I(k,G)^t$ es la entrada para las neuronas $w(k,i,j)^G$ y el canal azul $I(k,B)^t$ es la entrada para las neuronas $w(k,i,j)^B$. Luego que RESOM procesa los cuadros durante N_s iteraciones, los pesos resultantes son reordenados en un arreglo de cuatro imágenes $V_w(x,y,z)^{m,t}$, donde:

$$m = j + (i-1)J \tag{3.1}$$

Para este caso, J es la dimensión máxima de j, $m=1,..,4$. $V_w(x,y,z)^{m,t}$ se utiliza para definir $I_b(x,y,z)^t$ e $I_d(x,y)^t$; $I_b(x,y,z)^t$ es el valor esperado de $V_w(x,y,z)^{m,t}$, $I_d(x,y)^t$ es la suma de las diferencias de las imágenes en $V_w(x,y,z)^{m,t}$. Con esta información, $I_d(x,y)^t$ se utiliza para ubicar e inhibir información relacionada a los objetos dinámicos de $I_b(x,y,z)^t$, obteniendo en $I_b(x,y,z)^t$ el modelado de fondo. Finalmente, la segmentación de objetos dinámicos se realiza mediante substracción de fondo. En las siguientes subsecciones se explicarán cada una de las etapas de este método.

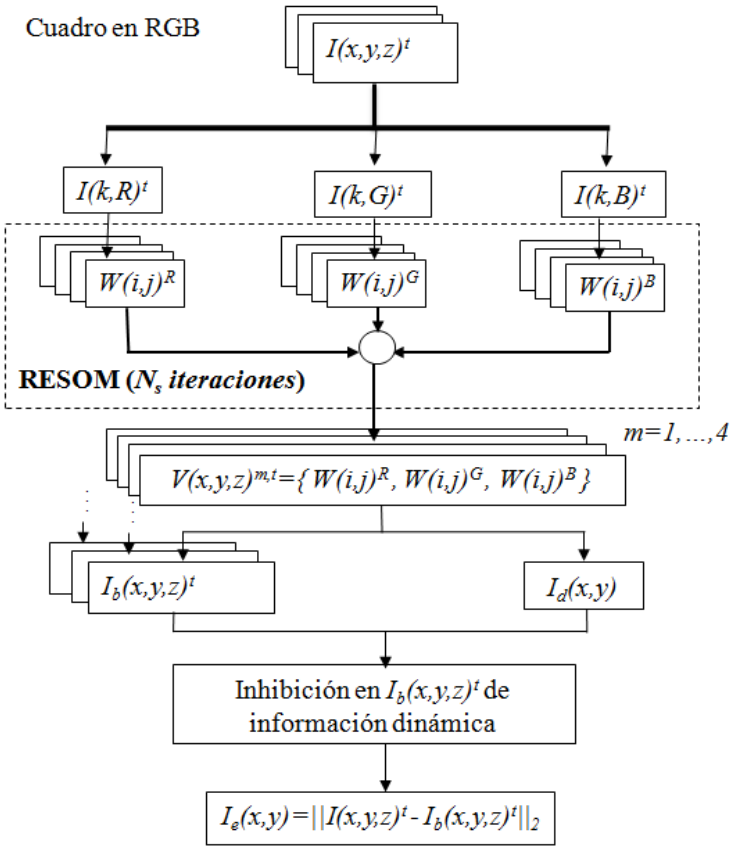


Figura 3.2 Diagrama del método basado en RESOM.

3.2.1 Funcionamiento Retinotópico de SOMRM y SOM *Kohonen*

La SOM de *Kohonen* es una red comúnmente utilizada en aplicaciones de procesamiento de imágenes y visión por computadora, y tiene una regla de competición basada en encontrar el mínimo de las distancias euclidianas entre la entrada y los pesos de la red. Si se utilizara este esquema en la SOMRM, esta regla se puede definir como:

$$W_{in} = \min_{i,j}\left(\sum_k \left\| I(k,z)^t - \omega(k,i,j)^z \right\|_2\right) \tag{3.2}$$

donde $W_{in}=(i_w,j_w)$. La regla de aprendizaje Hebbiana de la SOM de *Kohonen* tiene como elemento de actualización de pesos por iteración $\Delta\omega(k,i,j)^{n_s}$, la diferencia euclidiana entre los pesos y la entrada de la siguiente forma:

$$\begin{aligned}\omega(k,i,j)^{n_s+1} &= \omega(k,i,j)^{n_s} + \Delta\omega(k,i,j)^{n_s} \\ \Delta\omega(k,i,j)^{n_s} &= \alpha\left[I(k) - \omega(k,i,j)^{n_s} \right]\beta(i,j)\end{aligned} \tag{3.3}$$

Las ecuaciones 3.2 y 3.3 son en esencia, la competición y aprendizaje de la SOM de *Kohonen* [113], α es la taza de aprendizaje y $\beta(i,j)$ la función de vecindario. Un aspecto interesante es que si se combinan las ecuaciones 2.23 y 2.24 de SOMRM con las ecuaciones 3.2 y 3.3 de SOM *Kohonen*, es que se puede hacer una nueva versión de SOM con el mismo mecanismo retinotópico de aprendizaje y la misma arquitectura que la SOMRM original. Sin embargo, es necesario cuidar ciertos aspectos en el aprendizaje del escenario de la secuencia de video, ya que el modelo SOMRM se diseñó para entrenar pesos con patrones gaussianos. Si se analizan las ecuaciones 2.24, 2.25, 3.2 y 3.3, se deben de considerar los siguientes puntos:

1). Si se utiliza la competición de la ecuación 3.2 y el aprendizaje Hebbiano de la ecuación 3.3, no se podría definir el factor $\eta(i,j)$ del parámetro $\beta(i,j)$ de la regla Hebbiana. Por lo tanto, se tendría que eliminar el termino de $\beta(i,j)$.

2). La regla Hebbiana de la ecuación 2.24 está normalizada. Por lo tanto, los valores resultantes de los pesos $\omega(k,i,j)^{n_s}$ están en un intervalo de $[0,\varepsilon]$, donde $\varepsilon\approx 0$.

3). El modelo original de SOMRM dado por las ecuaciones 2.23 y 2.24 diverge en algunos videos. Esto se debe al factor $\eta(i,j)$ de $\beta(i,j)$.

Con base a estos puntos se puede observar que para escenarios reales no es recomendable utilizar el factor $\eta(i,j)$ de $\beta(i,j)$ ni utilizar la versión normalizada de la regla Hebbiana. En las Figuras 3.3b a 3.3d, se puede ver el proceso de aprendizaje de una neurona en los primeros 20 cuadros de un video con un escenario fijo de una SOMRM con 4 neuronas y sin el factor $\eta(i_w,j_w)$ en la ecuación 2.26. En las Figuras 3.3e a 3.3h, se puede ver el proceso de aprendizaje de una neurona en los primeros 10 cuadros con el mismo video pero utilizando SOM *Kohonen* con 4 neuronas. Además, de los experimentos mostrados en las Figuras 3.3, se realizaron otros donde las ecuaciones de SOMRM y SOM

Kohonen se combinaron. Una opción es utilizar la competición dada por la ecuación 2.23 con la regla Hebbiana de SOM *Kohonen* de la ecuación 3.3. A esta combinación se le denomina RET-SOM. En la Figura 3.4a a 3.4c se observan los resultados. Otra opción es utilizar la competición de *Kohonen* de la ecuación 3.2 con la regla normalizada de la ecuación 2.25; a esta red se le denominará SOM-RET. En la Figura 3.4d a 3.4f se observan los resultados.

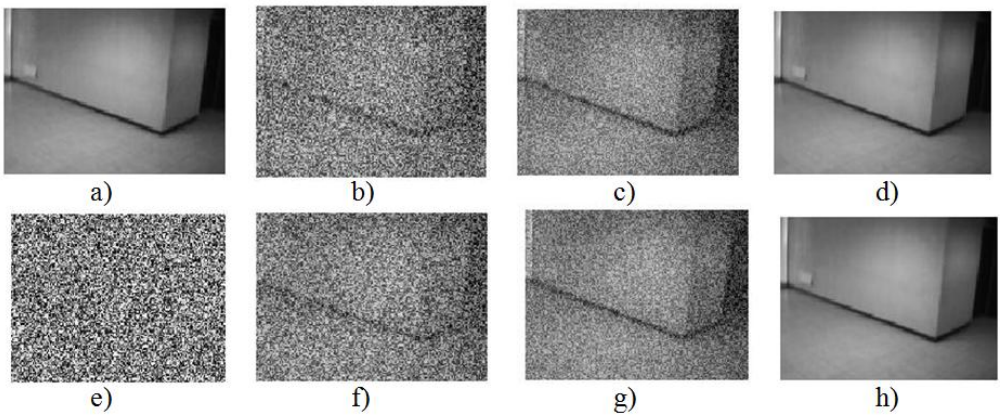


Figura 3.3. Resultados de procesamiento de SOM Retinotópica con las ecuaciones de competición y aprendizaje Hebbiano de SOMRM y SOM *Kohonen*. a). Escena de los primeros cuadros. b). $W(i_1,j_1)$ en $n_s=1$ con SOMRM. c). $W(i_1,j_1)$ en $n_s=2$ con SOMRM. d). $W(i_1,j_1)$en $n_s=10$ con SOMRM. e). Condición inicial de un mapa de pesos. f). $W(i_1,j_1)$ en $n_s=1$ con SOM *Kohonen*. g). $W(i_1,j_1)$ en $n_s=2$ con SOM *Kohonen*. h). $W(i_1,j_1)$ en $n_s=10$ con SOM *Kohonen*.

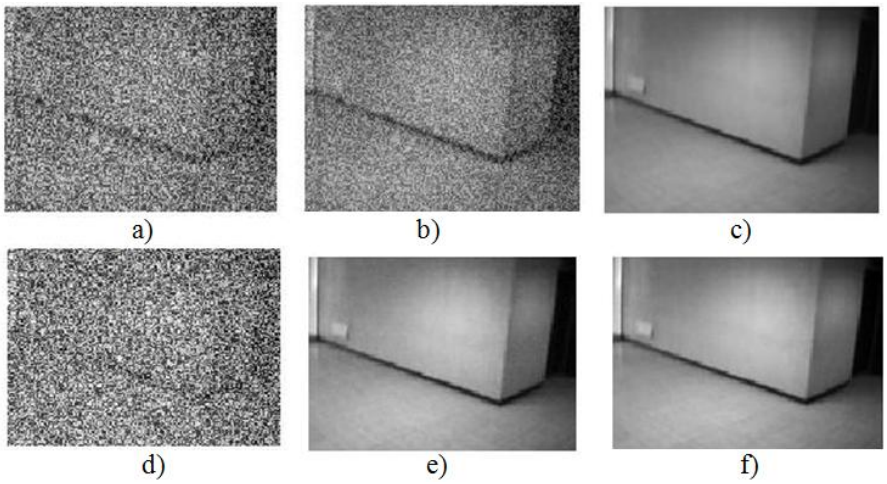


Figura 3.4 Resultados de combinaciones entre SOM *Kohonen* y SOMRM. a).-c). Salida de la red con la competición lineal con regla Hebbiana de *Kohonen* en $n_s=1$, $n_s=2$, $n_s=10$. d).-f). Salida de la red con la competición de *Kohonen* con regla Hebbiana normalizada en $n_s=1$, $n_s=2$, $n_s=10$.

Competición y aprendizaje Hebbiano de la SOM Retinotópica. En distintas pruebas realizadas, se observaron resultados muy similares en el aprendizaje de SOM *Kohonen*, SOMRM, RET-SOM y SOM-RET. En los experimentos con las cuatro variantes de SOM

con aprendizaje retinotópico analizados se pudo notar que la competición dada por la ecuación 3.2 genera los mismos resultados que la competición dada por la ecuación 2.24. Si se utiliza la ecuación 2.24 los valores de $\eta(i,j)$ serán muy similares ya que el escenario es el mismo y las neuronas aprenden lo mismo; por tanto, neuronas y escenario tienen la misma información, por lo que la selección de $W(i_w,j_w)$ en cada iteración es aleatoria. La ecuación 3.2 obtiene la neurona ganadora con una medición de similitud entre las neuronas $W(i,j)$ y el cuadro $I(k)^t$ utilizando la distancia euclidiana. En este caso, mientras el escenario no tenga cambios, las neuronas aprenderán la misma información, por lo que la selección de $W(i_w,j_w)$ es también aleatorio. Un aspecto interesante de la competición dada por la ecuación 2.24, es que esta regla se diseño como una función lineal que asemeja el comportamiento de células lineales en V1. El producto punto de las neuronas $W(i,j)$ con $I(k)^t$ se agrega linealmente igual que en redes como perceptrón. En este caso, se tendría una función de activación lineal y la respuesta está dada por:

$$\eta(i,j) = f\left(\sum_k I(k)^t \omega(k,i,j)\right) \tag{3.4}$$

En este caso, $\eta(i,j)=f(.)$ es simplemente multiplicar por uno el producto punto de la neurona $W(i,j)I(k)^t$. De esta manera, tal como se vio en las simulaciones de SOMRM y LISSOM, la $\eta(i,j)$ no solo es una competición entre neuronas, también es la respuesta de V1.

Entre las implementaciones de la regla Hebbiana dadas por la ecuación 2.25 o 3.3, se pudo observar que ambas funcionan de manera similar. La principal diferencia entre ambas es que la regla dada por la ecuación 2.25 aprende de manera más rápida, ya que $\Delta\omega(k,i,j)$ es directamente proporcional a $I(k)^t$, mientras que la ecuación 3.3 se basa en la distancia euclidiana entre $I(k)^t$ y los pesos $\omega(k,i,j)$, lo cual se puede apreciar en la Figura 3.4 donde el escenario fue aprendido en $n_s=2$ con la ecuación 2.25, mientras que en esa iteración la ecuación 3.3 aún no completaba el aprendizaje del escenario. No obstante, de acuerdo a la experimentación realizada no es recomendable utilizar la regla Hebbiana de la ecuación 2.25 por dos razones:

1). La respuesta normalizada genera resultados con valores de $[0,\varepsilon]$. Estos valores son poco prácticos si se va a utilizar un algoritmo con varias etapas aparte de SOM Retinotópica. Por

lo que se necesitaría una capa adicional que mapeara los valores de $[0,\varepsilon]$ a [0,1] antes de pasar a otras etapas en un algoritmo.

2). En la ecuación 2.25 $\Delta\omega(k,i,j)=\alpha\beta(i,j)I(k)$; aunque ciertamente este factor ayuda a que la regla Hebbiana aprenda más rápido un escenario, en 20 iteraciones, resulta ser una desventaja si se quiere que el escenario se aprenda en una o dos iteraciones. En una iteración n_s, la actualización de los pesos en ecuación 2.25 se expresa como:

$$\omega(k,i,j)^{n_s}+\alpha\beta(i,j)I(k) \tag{3.5}$$

En el caso de la ecuación 3.3, la actualización de pesos en una iteración se expresa como:

$$\omega(k,i,j)^{n_s}+\alpha\beta(i,j)\left[I(k)-\omega(k,i,j)^{n_s}\right] \tag{3.6}$$

Con base a las ecuaciones 3.5 y 3.6 se puede observar que si se necesita que la red neuronal aprenda el escenario en una iteración, la regla dada por SOM *Kohonen* es más rápida ya que si se analiza la ecuación 3.3, si en una iteración n_s, $\alpha\beta(i,j)=1$, el peso nuevo $\omega(k,i,j)^{n_s+1}$ es $I(k)$, mientras que en el caso de la ecuación 2.25, si en una iteración n_s, $\alpha\beta(i,j)=1$, el peso nuevo $\omega(k,i,j)^{n_s+1}$ es $I(k)+\omega(k,i,j)^{n_s}$. La razón por la que SOM-RET aprendió más rápido que RET-SOM es porque $\alpha<0.42$; sin embargo, si $\alpha>0.5$, RET-SOM aprende mucho más rápido que SOM-RET, ya que en RET-SOM se pueden eliminar valores de $\omega(k,i,j)^{n_s}$ anteriores por la diferencia dada en $\Delta\omega(k,i,j)$. Por lo tanto, con la ecuación 3.3 se puede controlar mejor el aprendizaje de la red neuronal, ya que con valores bajos de α se puede tener un aprendizaje lento, mientras que con valores altos se puede tener un aprendizaje muy rápido; incluso de un cuadro a otro.

Por tanto, de acuerdo a este análisis se puede concluir que en el diseño de SOM Retinotópica se puede recurrir a cualquier regla de competición (ecuación 2.24 o 3.2) ya que ambas se comportan de forma similar, pero para el aprendizaje Hebbiano es más recomendable utilizar la ecuación 3.3.

Función vecinal. Una parte importante en la SOM Retinotópica es la función vecinal $\beta(i,j)$, ya que es un parámetro que caracteriza la capacidad de aprendizaje a cada neurona en cada iteración. Esta caracterización es la que forma un mecanismo topográfico que se documentó en la sección 2.4.1. Es decir, $\beta(i,j)$ asigna a cada neurona en $W(i,j)$ cierta capacidad de actualización en cada iteración n_s. En la subsección 2.4.1 se reportó una función vecinal

proporcional a la respuesta de las neuronas $\eta(i_w,j_w)$ que da como salida valores de actualización altos y los pesos pueden divergir al utilizar escenarios reales. Una forma de eliminar este problema es retirando a la ecuación 2.26 el factor $\eta(i_w,j_w)$.

Para analizar cómo influye el factor $\eta(i_w,j_w)$ en el aprendizaje Hebbiano, se realizaron diversos experimentos. Uno de ellos consistió en seleccionar al azar algunos videos, los cuales fueron '*V2*', '*V1*', '*PT1b*' y '*CF*' (ver Figura 3.1). Los videos tienen una resolución a 120x160 (*K=19200*) y se corrieron a escala de grises, utilizando una RET-SOM de 4 neuronas (SOM-RET no fue tomada porque utiliza la regla Hebbiana de la ecuación 2.24). Además, se realizó un segundo experimento donde estos videos fueron recortados para tener otros videos de 24x24 (K=576). En ambos experimentos RET-SOM saturaba sus pesos desde la primer iteración; al obtener valores de $\eta(i_w,j_w)$ el promedio E_I de $\eta(i_w,j_w)$ para videos con resolución de 120x160 fue de $E_I=4445.31$. Para los videos con resolución de 24x24, el promedio fue de $E_I=163.37$. Estos valores tan altos son los causantes de la divergencia en la regla Hebbiana, ya que para todas las neuronas $\alpha\beta(i,j)>1$, por tanto, si $n_s\rightarrow\infty$, entonces:

$$\left|\sum_k \omega(k,i,j)\right| \rightarrow \infty \tag{3.7}$$

En el caso de los experimentos de la subsección 2.4.1 donde la retina en SOMRM es de 24x24, los pesos no divergen ya que se utiliza la regla Hebbiana normalizada y la mayor parte de los pixeles tienen valores de cero en $I(x,y)$. De acuerdo a los experimentos anteriores, el valor E_1 de $\eta(i_w,j_w)$ es más alto en los videos con resolución de 120x160 que el promedio en los videos de 24x24. Esto es debido al tamaño del video; ya que al aumentar la resolución espacial los valores de $\eta(i,j)$ tienden a aumentar a causa del producto punto de $\eta(i,j)=IW(i,j)$ de la ecuación 2.23. Por ejemplo, si se tuviera una imagen cuya distribución de niveles de grises tuviera valores entre [0,1] y formara una distribución de probabilidad gaussiana, entonces de acuerdo al teorema del límite central [114], el valor esperado para la imagen es $E_I=0.5$, mientras que el valor esperado para los pesos que aprenden la imagen es $E_\omega=0.5$. Ahora bien, dado que $\eta(i,j)$ está dado por el producto punto de la ecuación 2.23, entonces, tomando como base los valores esperados, este parámetro se puede replantear como:

$$\eta(i,j) \approx KE_I E_\omega \approx 0.25K \tag{3.8}$$

Con base a la ecuación 3.8, se espera que $\eta(i,j) \approx 144$ para un cuadro con resolución de 24x24; para un cuadro con resolución de 120x160 $\eta(i,j) \approx 4800$. Como se puede notar, los valores obtenidos por la ecuación 3.8 son similares a los obtenidos en $\eta(i,j)$. En diversas pruebas realizadas con videos en diferentes resoluciones, los resultados de $\eta(i,j)$ fueron consistentes con los resultados de la ecuación 3.8, aun cuando muchos videos tienen escenarios con distribuciones de niveles de grises multimodales. En la Tabla 3.1 se pueden observar varios ejemplos de videos con distintas resoluciones, el valor resultante de la ecuación 3.8 y el valor de $\eta(i_w,j_w)$ para cada video. Los videos son '*V2*' y '*PT1B*' y son recortados para bajar la resolución. En todos los casos, los resultados de $\eta(i,j)$ tienen consistencia con los valores de la ecuación 3.8.

Tabla 3.1. Resultados de $\eta(i_w,j_w)$ para distintas resoluciones

Resolución	Video	ecuación 3.8	ejemplo de $\eta(i,j)$
24x24	*PT1b*	144	130
24x24	*V2*	144	148
50x50	*PT1b*	625	601
50x50	*V2*	625	605
100x100	*PT1b*	2500	2005
100x100	*V2*	2500	2589
120x160	*PT1b*	4800	4005
120x160	*V2*	4800	4395

Con base a la ecuación 3.8 y los experimentos planteados, se puede concluir que existe una relación entre el valor de $\eta(i,j)$ y la resolución espacial de la imagen; por tanto, es mejor retirar este factor de la función vecinal $\beta(i,j)$. No obstante, un aspecto que se debe tomar en cuenta es si este factor se debe considerar en el aprendizaje de SOMRM. Al analizar este modelo en [34], se puede concluir que la función vecinal $\beta(i,j)$ es la que define el aprendizaje; las diferencias de valores de $\beta(i,j)$ se modelan a partir de σ_β, la distancia entre el índice de $W(i,j)$ y $W(i_w,j_w)$. $\eta(i_w,j_w)$ se utiliza solo para modificar la amplitud de $\beta(i,j)$ y dar mayor influencia de la actividad de la neurona ganadora en el resto del vecindario. Dado que $\eta(i_w,j_w)$ solo modifica la amplitud de $\beta(i,j)$, y no modifica el aprendizaje, entonces la función vecinal para la SOM Retinotópica en esta tesis estará dada por:

$$\beta(i,j) = \exp\left(-\frac{(i-i_w)^2 + (j-j_w)^2}{\sigma_\beta^2}\right) \tag{3.9}$$

Con la ecuación 3.9, se pueden preservar las propiedades para realizar una SOM Retinotópica.

En conclusión, si se utiliza la ecuación 3.9 para $\beta(i,j)$, la regla Hebbiana de la ecuación 3.3 y cualquiera de las dos competiciones (ecuación 2.24 o 3.2), se puede obtener una red SOM con principios de retinotopía para analizar videos con escenarios reales.

3.2.2 Diseño de Red SOM Retinotópica (RESOM)

Basándose en los experimentos de la subsección anterior, la sección 2.4 donde se documenta la SOMRM, y en la consideración que un video real es un conjunto de cuadros a color de tamaño no conocido, se pueden asumir las siguientes consideraciones:

- Para efectos de entrenamiento, en SOMRM si $n_s \rightarrow \infty$, $\alpha=0$. Sin embargo, en un escenario real en una secuencia de video siempre existen cambios, por lo tanto, α debe ser distinta de 0.
- De acuerdo a la subsección 3.2.1, es posible utilizar cualquier competición (ecuación 2.23 o 3.2) para analizar videos reales.
- La regla Hebbiana debe basarse en la ecuación 3.3.
- La función vecinal debe basarse en la ecuación 3.9.
- La SOM Retinotópica se utilizará en videos a color; cada cuadro es una imagen en RGB $I(x,y,z)^t$. Para este trabajo, cada canal de color z se procesa con su propia red.
- Para facilitar los cálculos computacionales, los índices (x,y) se transforman a k (ecuación 2.30) para las neuronas se utilizará el índice m (ecuación 3.1).

Con base a estos argumentos, el modelo de la SOM Retinotópica para análisis de video (RESOM) queda definido de la siguiente forma; la entrada es un cuadro $I(x,y,z)^t$ donde se transforma a $I(k,z)^t$. La competición de la RESOM va a estar dada por:

$$\eta(m)^{z,n} = \sum_k I(k,z)^t \omega(k,m)^{z,n} = I^t W(m)^{z,n} \tag{3.10}$$

La respuesta neuronal $\eta(m_w)^z$ con mayor magnitud en cada vecindario de color es considerada la ganadora, n es el índice de iteraciones de la RESOM. Adicionalmente, como se mencionó en la subsección 3.2.1, se puede utilizar una competición basada en SOM

Kohonen y se basa en la sumatoria de la distancia entre los pesos de $W(m)^z$ de cada canal *z* de color y la entrada, lo cual está dado por:

$$\eta(m)^{z,n} = \sum_k \left| I(k,z)^t - \omega(k,m)^{z,n} \right| \tag{3.11}$$

En este caso la respuesta neuronal en $\eta(m)^z$ con menor magnitud en cada vecindario de color indica la neurona ganadora $W(m_w)^{z,n}$. La iteración $n=1$ de RESOM inicia con $t=1$ y continua iterando a N_s veces por cuadro hasta que termina la secuencia de video. Aunque se realizaron diferentes trabajos utilizando la competición de ecuación 3.10, el diseño del método SOM-CNN se realizó con la competición dada por la ecuación 3.11. Sin embargo, en [115] se muestra un análisis de la RESOM con la regla 3.10. Sin importar cual regla será utilizada para la competición, el modelo que se describe se le denomina RESOM ya que el comportamiento es el mismo. Los pesos se van modelando con el aprendizaje Hebbiano que está dado por:

$$\omega(k,m)^{z,n+1} = \omega(k,m)^{z,n} + \Delta\omega(k,m)^{z,n} \tag{3.12}$$

$$\Delta\omega(k,m)^{z,n} = \alpha\left[I(k,z) - \omega(k,m)^{z,n} \right]\beta(m)^z \tag{3.13}$$

donde α es la taza de aprendizaje, $\beta(m)^z$ es la función vecinal. Ambos parámetros están basados en los parámetros Hebbianos de la SOMRM en [34], y en RESOM están dados por:

$$\alpha = \max\left[\exp\left(-\frac{6t_a}{T_f} \right), \alpha_s \right] \tag{3.14}$$

donde T_f es una constante que define la caída de la exponencial de la ecuación 3.14, en este trabajo se eligió que $T_f=35$ (más adelante se definirá porque $T_f=35$), α_s es una constante que no permite que α caiga a cero cuando $t_a \to \infty$. Inicialmente t_a es un índice auxiliar de cuadros que se actualiza con $t_a=t_a+1$ en cada cuadro y su condición inicial es $t_a=0$ (posteriormente se abordará el uso). La función vecinal caracteriza el aprendizaje retinotópico de los pesos de las neuronas $W(m)^z$, y está dada por:

$$\beta(m)^z = \exp\left(-\frac{\left(m_w^z - m \right)^2}{\sigma_\beta^2} \right) \tag{3.15}$$

donde (m_w) es la coordenada de la neurona ganadora del canal *z*, σ_β es el radio de vecindario dado por:

$$\sigma_\beta = \max\left[13.5\exp\left(-\frac{5t_a}{T_f}\right), \sigma_s\right] \tag{3.16}$$

donde σ_s no permite que σ_β caiga a cero cuando $t_a \to \infty$. Con este modelo, la RESOM es básicamente una SOM *Kohonen* con el mecanismo de aprendizaje de la SOMRM. Los parámetros de aprendizaje σ_β y α tienen los valores iguales para las tres redes. Los pesos de las neuronas tienen valores positivos en el intervalo de [0,1]. La arquitectura de la RESOM se puede observar en la Figura 3.5, donde el cuadro $I(x,y,z)^t$ se transforma a $I(k,z)^t$ que es la entrada a un conjunto de neuronas $W(i,j)^{z,n}$, de forma que el canal rojo $I(k,R)^t$ es la entrada al conjunto de neuronas $W(m)^{R,n}$, el canal verde $I(k,G)^t$ es la entrada al conjunto de neuronas $W(m)^{G,n}$ y el canal azul $I(k,B)^t$ es la entrada al conjunto de neuronas $W(m)^{B,n}$. Por tanto, RESOM tiene tres redes con *M* neuronas. RESOM puede utilizar cualquier cantidad de neuronas, aunque, por cuestión de costos computacionales, lo recomendable es recurrir a cantidades pequeñas de neuronas, en este trabajo se eligió que *M=4*.

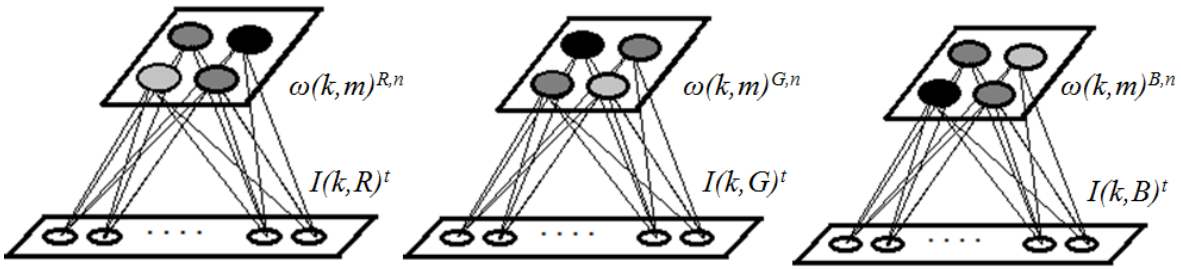


Figura 3.5 Arquitectura de la RESOM. La red consta de tres vecindarios para procesar con M neuronas cada uno de los canales de color.

En condiciones iniciales, los pesos son aleatorios y en cada iteración *n* primero se realiza la competición dada por la ecuación 3.11, donde la neurona ganadora es la actividad con menor magnitud en $\eta(m)^z$. Luego se calcula σ_β dada por la ecuación 3.16 y $\beta(m)^z$ dada por la ecuación 3.15. Una vez estimada la neurona ganadora y funciones vecinales, se calcula la taza de aprendizaje dada por la ecuación 3.14 y finalmente se actualizan los pesos con la ecuación 3.12. De esta manera, RESOM realiza este proceso de competición y actualización de pesos en cada iteración. RESOM procesa N_s iteraciones en cada cuadro. En la Figura 3.6 se puede observar el resultado de procesar una RESOM con *M=6*, donde los

pesos de las neuronas se reordenaron para formar 6 imágenes en RGB que muestran los patrones aprendidos por la red, mediante la siguiente relación:

$$V_w(k,z)^{m,t} = \omega(k,m)^{z,n} \tag{3.17}$$

Luego, $V_w(k,z)^{m,t}$ se transforma a $V_w(x,y,z)^{m,t}$. En la ecuación 3.17, para $V_w(k,z)^{m,t+1}$, se debe cumplir que $\omega(k,m)^{z,n+N_s}$.

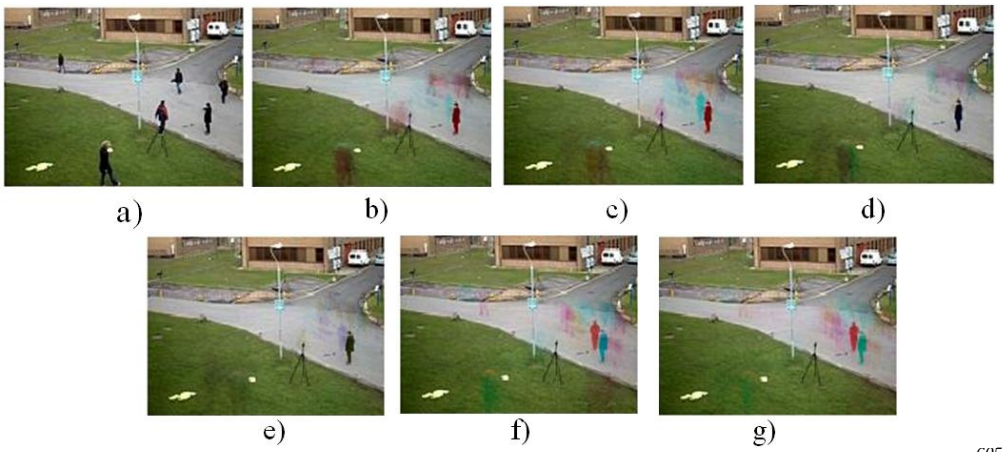


Figura 3.6. Respuesta de la RESOM con $M=6$, $N_s=1$, $\alpha_s=0.4$ y $\sigma_s=0.5$. a) Cuadro $I(x,y,z)^{605}$. b) $V_w(x,y,z)^{1}$. c) $V_w(x,y,z)^{2}$. d) $V_w(x,y,z)^{3}$. e) $V_w(x,y,z)^{4}$. f) $V_w(x,y,z)^{5}$. g) $V_w(x,y,z)^{6}$.

Aprendizaje de la RESOM. La RESOM va a tener tres parámetros de aprendizaje: α que define la capacidad de aprendizaje de todas las neuronas; si $\alpha \rightarrow 0$ la capacidad de aprendizaje decrece, pero si $\alpha > 1$, la red satura los pesos. Otro parámetro de aprendizaje es $\beta(m)^z$, y define a cada neurona cierta capacidad de aprendizaje con base a la distancia entre cada neurona con respecto a la neurona ganadora $W(m_w)^{z,n}$ y el radio de vecindario σ_β, de tal forma que $\beta(m)^z$ crece exponencialmente al aumentar el valor de σ_β. Finalmente, N_s define la capacidad de aprendizaje por cuadro; si $N_s \rightarrow \infty$, todas las neuronas aprenderán completamente la escena presente en un cuadro.

Para entender el comportamiento de la RESOM, de acuerdo a la ecuación 3.12, si $\beta(m)^z = 1$ en todas las iteraciones, los pesos se actualizan adaptándose a la información de cada cuadro, los pesos en cada cuadro $I(k,z)^{t+1}$se actualizan como:

$$\omega(k,m)^{z,n+N_s} = (1-\alpha)^{n+N_s}\,\omega(k,m)^{z,n} + \left[1-(1-\alpha)^{n+N_s}\right] I(k,z)^t \tag{3.18}$$

donde $\omega(k,m)^{z,n}$ son los pesos del cuadro anterior. Se puede observar en este caso que conforme pasan las iteraciones el valor de los pesos $\omega(k,m)^{z,n}$ converge al cuadro actual, y

conforme α disminuye, se necesitan más iteraciones para que $\omega(k,m)^{z,N_s}$ logre converger al cuadro actual. Este comportamiento al estar influido por $\beta(m)^z$ se define como:

$$\omega(k,m)^{z,n+N_s} = \left(1-\beta(m)_1^z\alpha\right)...\left(1-\beta(m)_{N_s}^z\alpha\right)\omega(k,m)^{z,n} + \left[1-\left[\left(1-\beta(m)_1^z\alpha\right)...\left(1-\beta(m)_{N_s}^z\alpha\right)\right]\right]I\left(k,z\right)^t \quad (3.19)$$

De acuerdo a la ecuación 3.12, si $\alpha=1$ y $N_s=1$, los pesos $\omega(k,m_w)^{z,n+N_s}$ de la neurona ganadora aprenderán por completo $I(k,z)^{t+1}$. Para el resto de las neuronas, los pesos se dan por $\omega(k,m)^{z,n+N_s} = [1-\beta(m)^z]\omega(k,m)^{z,n}+\beta(m)^z I(k,z)^t$. Según la ecuación 3.19, si $0 < \alpha < 1$ los pesos de la RESOM combinan información de $I(k,z)^t$ y valores pasados de $\omega(k,m)^{z,n}$, de tal forma que si N_s tiene un valor finito, entonces las redes aprenderán los objetos que permanecen en la misma posición durante varios cuadros (fondo) formando los mismos patrones en todas las neuronas. En el caso de los objetos que cambian de posición (dinámicos), estos serán aprendidos de forma incompleta y de acuerdo a los valores de $\beta(m)^z$, estos pueden aprenderse de forma muy diferente en cada neurona. De esta forma, si se cumplen las condiciones de $0 < \alpha < 1$ y N_s tiene un valor finito, los objetos dinámicos generan información que es distinta en todas las neuronas, mientras que el fondo genera la misma información en todas las neuronas, como el ejemplo de la Figura 3.6.

Respecto al aprendizaje, es importante que en el primer cuadro RESOM desarrolle un aprendizaje donde los pesos de las neuronas puedan aprender la escena. De acuerdo a las ecuaciones 3.14 y 3.16, en condiciones iniciales $\alpha=1$ y $\sigma_\beta=13.5$ (si $\sigma_\beta=13.5$ y $M=4$, entonces los valores en $\beta(m)^z$ serán similares a uno) y por lo tanto, todas las neuronas aprenden $I(k,z)^t$. Para garantizar el correcto aprendizaje de la escena en las neuronas, $N_s=3$ en el primer cuadro. No obstante, después RESOM ajusta el aprendizaje para seguir correctamente el fondo, para ello, $N_s=1$, mientras que σ_β y α empiezan a caer a σ_s y α_s, esto con el objetivo de que RESOM aprenda solo el fondo de una secuencia de video, mientras que los objetos dinámicos sean aprendidos en las neuronas ganadoras y de forma incompleta. De esta manera, conforme la secuencia de video pasa, los objetos dinámicos se aprenden en forma incompleta y con diferente patrón en cada neurona, mientras que el fondo se aprende con el mismo patrón en todas las neuronas. En la sección 3.4 se discutirán los valores de σ_s y α_s.

3.2.3 Método de Modelado de Fondo DR-SOM

Después del primer cuadro, los pesos aprenden la escena y el valor esperado de $V_w(x,y,z)^{m,t}$ dado por:

$$I_b(x,y,z)^t = \frac{1}{M}\sum_{m=1}^{M} V_w(x,y,z)^{m,t}, \qquad m = 1,..,M \tag{3.20}$$

se considera el modelado de fondo inicial. Después del primer cuadro, los parámetros de aprendizaje de la RESOM se adaptan para aprender el fondo de forma correcta en todas las neuronas y para que los objetos dinámicos sean aprendidos de manera diferente en cada neurona, como lo muestra en la Figura 3.6. De esta forma, $Ib(x,y,z)^t$ es un semi-modelado de fondo, ya que aun contiene información incompleta de los objetos dinámicos como en la Figura 3.7b y 3.7d.

a) b) c) d)

Figura 3.7. Valor esperado de $V_w(x,y,z)^{m,t}$. a) Video '*p2009_L1_1*', *t=180*. b) $Ib(x,y,z)^{180}$ de a). c) Video '*PT1B*' ,*t=140*. d) $Ib(x,y,z)^{60}$ de c).

Para reducir la información de objetos en movimiento en $I_b(x,y,z)^t$, se tomaron dos consideraciones: primero, si $I_b(x,y,z)^t$ es actualizada con una neurona de cada red por iteración, en vez de todas las neuronas, esta imagen puede ser menos sensible a cambios entre cuadros (como objetos en movimiento o ruido). Para ello, una aparente solución podría ser actualizar con la regla Hebbiana (ecuación 3.12) a una neurona por iteración en cada red. No obstante, en realidad esta solución no es efectiva ya que todas las neuronas deben ser actualizadas con la ecuación 3.12 en cada iteración, porque de otra forma estas pueden empezar a mantener información del fondo de varios cuadros pasados debido a que las neuronas pueden no actualizarse cuando sean ganadoras. Otra posible solución es actualizar a una neurona por red y por iteración en $V_w(x,y,z)^{m,t}$, en vez de todas como en la ecuación 3.17. De esta manera, en el método de modelado de fondo, para el primer cuadro $V_w(x,y,z)^{m,t}$ se define por la ecuación 3.17, pero a partir del segundo cuadro, $V_w(x,y,z)^{m,t}$ se puede definir por:

$$V_w(k,z)^{m,t+1} = \begin{cases} w(k,m)^{z,n+N_s} & si \quad m_r = m \\ V_w(k,z)^{m,t} & otra \quad forma \end{cases} \tag{3.21}$$

donde m_r es el residuo de t/M. Luego $V_w(k,z)^{m,t+1}$ se transforma a $V_w(x,y,z)^{m,t+1}$. Con las ecuaciones 3.17 y 3.21, la cantidad de neuronas para actualizar $V_w(x,y,z)^{m,t}$ se manejan como un parámetro denominado v_w en DR-SOM; si se utiliza la ecuación 3.17, entonces v_w=4 (se actualiza $V_w(x,y,z)^{m,t}$ con 4 neuronas por cuadro en cada red z). Si se utiliza la ecuación 3.21, entonces v_w=1 (se actualiza $V_w(x,y,z)^{m,t}$ con 1 neurona por cuadro en cada red z). Al igual que N_s, v_w define la capacidad de aprendizaje por cuadro, pero con la diferencia que al aumentar N_s aumenta el tiempo de procesamiento, v_w mantiene los mismos tiempos de procesamiento sin importar cuál sea la ecuación seleccionada.

La segunda consideración se basa en lo siguiente; RESOM tiende a aprender los objetos de fondo en secuencias de video de forma correcta en todas las neuronas, mientras que los objetos en movimiento se aprenden en forma incompleta y distinta en cada neurona como se ve en la Figura 3.6. Esto se debe a que solo las neuronas ganadoras en cada iteración n aprenden los objetos en movimiento de forma parcial, mientras que el resto de las neuronas no aprenden los objetos en movimiento (los valores de los parámetros de aprendizaje se analizarán posteriormente). Dado que los objetos dinámicos no permanecen fijos, la información parcial aprendida en las neuronas en iteraciones anteriores se va cambiando por información del fondo. Por tanto, como se mencionó anteriormente, se asume que la información similar en todas las neuronas corresponde a los objetos estáticos, mientras que la información que es distinta en todas las neuronas se forma a partir de los objetos dinámicos. Entonces, se diseñó un método que encuentra las diferencias entre las neuronas de la RESOM para inhibirlas de $I_b(x,y,z)^t$. Para este método, primero se define la cantidad de diferencias $M_1=\lceil (M/2)^2 \rceil$ que se van a realizar, luego se define un arreglo con un conjunto de imágenes que contienen la diferencia absoluta entre todas las imágenes del arreglo $V_w(x,y,z)^{m,t}$, lo cual está dado por:

$$\begin{gathered} IV(x,y,z,m_k) = \left| V_w(x,y,z)^{m_1} - V_w(x,y,z)^{m_2} \right| \\ m_k = 1,2,...,M_1 \end{gathered} \tag{3.22}$$

donde $m_1 = m_k - A_v M/2$, $m_2 = M/2 + A_v + 1$. Inicialmente, $A_v=0$, y se actualiza con la siguiente regla:

$$A_v = \begin{cases} A_v + 1 & \text{Re}\,siduo(M_1 / m_k) = 1 \\ A_v & otra \quad forma \end{cases} \tag{3.23}$$

Con el objetivo de obtener estas diferencias en una imagen, se define la siguiente suma:

$$I_d(x,y)^t = \sum_{m_k} \left(\frac{1}{3} \sum_z IV(x,y,z,m_k) \right) \tag{3.24}$$

El resultado es una imagen que resume las diferencias en $V_w(x,y,z)^{m,t}$ como en la Figura 3.8b. Luego, $I_d(x,y)^t$ se utiliza para encontrar en $I_b(x,y,z)^t$ los pixeles contaminados para eliminar la información parcial derivada de los objetos dinámicos en $I_b(x,y,z)^t$ con la siguiente regla:

$$I_b(x,y,z)^{t+1} = \begin{cases} I_b(x,y,z)^{t+1} & I_d(x,y)^{t+1} < Th_1 \\ I_b(x,y,z)^t & otra \quad forma \end{cases} \tag{3.25}$$

donde Th_1 debe ser cero. No obstante, tal como en las Figuras 3.8b y 3.9b muestran, $I_d(x,y)^t$ puede retener ruido e información pasada de los objetos dinámicos. Por tanto, se realizó un análisis de la distribución de los niveles de grises de $I_d(x,y)^t$ en esas situaciones, con el cual se estableció un umbral en $Th_1=0.02$ para la ecuación 3.25. Una vez implementada la ecuación 3.21, $I_b(x,y,z)^t$ se puede utilizar como una imagen que contiene el modelado de fondo, el cual se utiliza para realizar la substracción de fondo [104], como se muestra en la Figura 3.8c. En este trabajo, la substracción se basa en la distancia euclidiana entre $I(x,y,z)^t$ y $I_b(x,y,z)^t$ dada por:

$$I_e(x,y)^t = \left\| I(x,y,z)^t - I_b(x,y,z)^t \right\|_2 \tag{3.26}$$

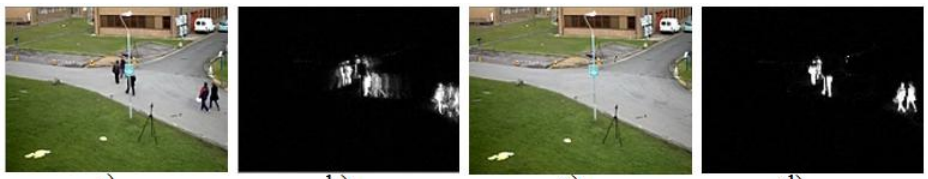


Figura 3.8. Procesamiento de DR-SOM '*p2009_L1_1*' después de una condición de *bootstraping*. a) $I(x,y,z)^{24}$. b) $I_d(x,y)^t$. c) $I_b(x,y,z)^t$ después de la ecuación 3.25. d) $I_e(x,y)^t$.

El intervalo de $I_e(x,y)^t$ es [0,1], donde un pixel $\rho(x,y) \epsilon I_e(x,y)^t$ con valor de cero implica que en ese punto no hay movimiento. Como se muestra en la Figura 3.2, el método DR-SOM termina cuando se obtiene $I_e(x,y)^t$, el cual se puede interpretar como una

segmentación suave de movimiento. Es decir, $I_e(x,y)^t$ se puede interpretar como la alerta visual que indica los puntos de atención del escenario donde existe movimiento.

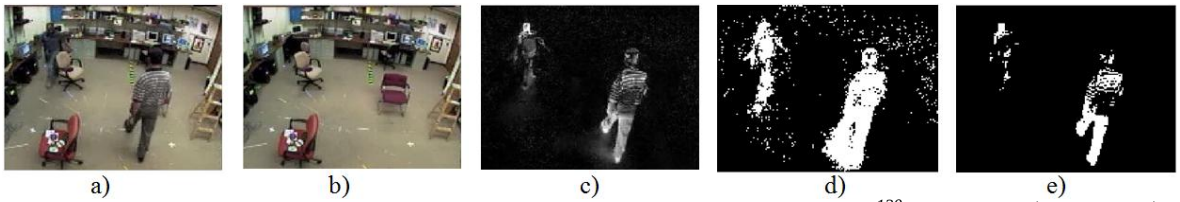
a) b) c) d)

Figura 3.9. Procesamiento de DR-SOM en video '*Vi1*'. a) $I(x,y,z)^{130}$. b) $I_d(x,y)^t$. c) $I_b(x,y,z)^t$ después de la ecuación 3.25. d) $I_e(x,y)^t$.

El resultado esperado en la mayor parte de las aplicaciones relacionadas con segmentación de objetos dinámicos es una segmentación dura. Es decir, una vez obtenido el resultado de las distancias euclidianas $I_e(x,y)^t$, el siguiente paso es binarizar el resultado. En las Figuras 3.10 y 3.11 se observan algunos resultados de segmentación dura basada en el método tradicional de umbralización; con base a un umbral Th_2 se define si cada pixel en $I_e(x,y)^t$ pertenece al Fondo o contiene información del objeto dinámico.

a) b) c) d) e)

Figura 3.10. Resultados de segmentación de DR-SOM con video '*P2001_1*'. Los objetos dinámicos están resaltados dentro de un circulo negro. a). $I(x,y,z)^{480}$. b). $I_b(x,y,z)^t$. c). $I_e(x,y)^t$. d). Umbralización de $I_e(x,y)^t$ con $Th_2=0.1$. e). Umbralización de $I_e(x,y)^t$ con $Th_2=0.35$.



a) b) c) d) e)

Figura 3.11. Resultados de segmentación de DR-SOM con video '*PV*'. a). $I(x,y,z)^{120}$. b). $I_b(x,y,z)^t$. c). $I_e(x,y)^t$. d). Umbralización de $I_e(x,y)^t$ con $Th_2=0.1$. e). Umbralización de $I_e(x,y)^t$ con $Th_2=0.35$.

Como se puede observar en las Figuras 3.10d y 3.10e y las Figuras 3.11d y 3.11e, los resultados en la segmentación dura son muy sensibles a la selección de Th_2, debido a que aún cuando DR-SOM obtuvo para ambos casos el mismo $I_e(x,y)^t$, la segmentación dura es muy distintas. Incluso, en las Figuras 3.10d y 3.11d se puede observar que un umbral bajo causa problemas de falsos positivos, sin embargo, en las Figuras 3.10e y 3.11e se puede observar que un umbral medio causa problemas de falsos negativos. En un método de

segmentación de objetos dinámicos, es importante tener un algoritmo que seleccione correctamente los criterios para lograr una segmentación dura aprovechando de forma adecuada la salida del modelado de Fondo. En este trabajo de tesis, se recurrió a una Red Neuronal Celular para esta segmentación.

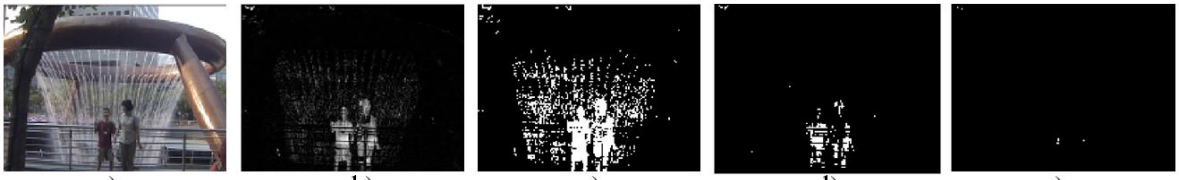
3.3 Binarización mediante Red Neuronal Celular

Como se mencionó previamente, el resultado de DR-SOM es una segmentación suave en $I_e(x,y)^t$. Por lo tanto, el siguiente paso para lograr la segmentación de objetos dinámicos es un proceso de segmentación dura o binarización que seleccione correctamente los criterios que definan las regiones para representar los objetos dinámicos. Una aparente solución sería realizar un método de umbralización tradicional que seleccione en forma automática Th_2 y genere una imagen $I_{eb}(x,y)^t$, la cual es una versión binarizada de $I_e(x,y)^t$. Sin embargo, este método no es una forma práctica de realizar la binarización ya que $I_e(x,y)^t$ genera regiones no solo de objetos dinámicos, también genera regiones causadas por diferentes tipos de ruido.

En la Figura 3.12a, se observa el cuadro *t=178* del video '*FT*' en el cual aparecen dos personas como objetos dinámicos y atrás de ellos, una cortina de agua, la cual forma un fondo dinámico. En la Figura 3.12c, se muestra $I_{eb}(x,y)^t$ con un umbral Th_2 de 0.1, donde se aprecia que en los resultados, se presenta tanto los objetos dinámicos como ruido por falsos positivos debido al fondo dinámico. Con $Th_2 = 0.5$ se obtiene el resultado que se muestra en la Figura 3.12d, donde se aprecia que se obtienen solo los objetos dinámicos, y con $Th_2 = 0.85$, ni los objetos dinámicos ni el fondo dinámico se muestran. En este ejemplo se ilustra un problema ya mencionado; si se pone un umbral bajo, el método es tan sensible que se detectan objetos dinámicos falsos, si se pone un umbral alto, el método pierde capacidad en detectar objetos dinámicos. Aunque en este ejemplo con un umbral intermedio se puede realizar una segmentación coherente[16], no en todos los demás ejemplos es así, ya que la definición de bajo o alto umbral es distinta en cada caso. Como ejemplo, en la segmentación del video '*CAM*' (Figura 3.13), la umbralización con $Th_2 = 0.1$ (Figura 3.13c) y $Th_2 = 0.5$ (Figura 3.13d) genera resultados regulares ya que el ruido sigue

[16] Segmentación coherente se refiere a que las regiones resultantes que representan los objetos dinámicos tienen características geométricas similares al objeto.

formando problemas de fondos dinámicos. Además, el resultado de segmentación del video '*CAM*' con $Th_2 = 0.85$ (Figura 3.13e) pierde el objeto dinámico.



a) b) c) d) e)

Figura 3.12. Resultados de umbralización de video '*FT*'. a). Cuadro $t=179$. b). $I_e(x,y)^t$. c). Umbralización de $I_e(x,y)^t$ con $Th_2=0.1$. d). Umbralización de $I_e(x,y)^t$ con $Th_2=0.5$. e). $I_e(x,y)^t$ con $Th_3=0.85$.

a) b) c) d) e)

Figura 3.13. Resultados de umbralización de video '*CAM*'. a). Cuadro $t=35$. b). $I_e(x,y)^t$. c). $I_{eb}(x,y)^t$ con $Th_2=0.1$. d). $I_{eb}(x,y)^t$ con $Th_2=0.5$. e) $I_{eb}(x,y)^t$ con $Th_2=0.85$.

En las Figuras 3.12 y 3.13 se ilustra que por un lado se pueden definir parámetros para segmentar correctamente un objeto dinámico, pero con altos niveles de ruido. Por otro lado, si se desea obtener un buen rechazo al ruido, se pierde información útil en la segmentación. Por lo cual, es necesario buscar estrategias de binarización que reduzcan problemas derivados de la relación falsos positivos/falsos negativos. Para buscar alguna estrategia de binarización, se tomó la siguiente consideración: el ruido causado por problemas como fondos dinámicos, ruido de video, y en ocasiones movimiento de cámara causa múltiples regiones con áreas irregulares, mientras que los objetos dinámicos forman regiones con áreas mayores al ruido. En las Figuras 3.12b y 3.13b se puede observar ejemplos de las regiones en $I_e(x,y)^t$ formadas por ruido derivado de fondos dinámicos. A partir de esto, se define que cualquier pixel con valor cercano a uno, pero cuyos vecinos tienen valores cercanos a cero es considerado ruido. Esta definición es el punto de partida para proponer un método de binarización basado en Redes Neuronales Celulares.

3.3.1 Red Neuronal Celular de Binarización por Vecindario.

Basándose en la consideración anterior, se asumió en esta tesis que se debe buscar un método de umbralización que vaya generando en cada pixel de $I_e(x,y)^t$ un umbral que dependa de la información del vecindario. Existen varias alternativas para realizar algoritmos de segmentación y preprocesamiento, y un modelo ampliamente utilizado para

desarrollar métodos para estas tareas es la Redes Neuronales Celular (CNN) [116]. Este modelo permite una interacción de vecinos de cada unidad en la red. Cada unidad es un elemento de procesamiento no lineal, normado y disipativo[17] llamado célula [117]. Aunque esta red surge a partir de implementar un modelo analógico, existe la CNN de tiempo continuo (CTCNN) y la CNN de tiempo discreto (DTCNN). Esta última es básicamente una CTCNN expresada en términos de ecuaciones de diferencias (la CTCNN esta descrita en términos de ecuaciones diferenciales). A grandes rasgos la DTCNN está dada por:

$$\chi(i,j)^{n_c} = \sum_{(i_N,j_N)\in N_v} A(i_N,j_N)y_c(i+i_N,j+j_N)^{n_c} + \sum_{(i_N,j_N)\in N_v} B(i_N,j_N)I(i+i_N,j+j_N)^{n_c} + Z_c(i,j) \qquad (3.27)$$

donde n_c son las iteraciones de la red, N_c es el vecindario del elemento $\rho(i,j)\epsilon I(i,j)$, (i_N,j_N) son índices del vecindario N_c, $\chi(i,j)$ es el estado de cada elemento de la CNN, A y B son las plantillas de pesos de la red, $Z_c(i,j)$ es un umbral, $y_c(i,j)$ es la salida dada por:

$$y_c(i,j)^{n_c+1} = \frac{1}{2}\left(\left|\chi(i,j)^{n_c}+1\right| - \left|\chi(i,j)^{n_c}-1\right|\right) = f\left[\chi(i,j)\right] \qquad (3.28)$$

En [116] se muestran varios modelos basados en CNN para distintos propósitos como detección de bordes, esquinas, movimiento, binarización, etc. Estos modelos parten de la CNN y las diferencias entre ellos están principalmente en las plantillas A, B y el umbral $Z_c(i,j)$. Entre estos modelos, la CNN de Binarización (TCNN) es un método iterativo con las siguientes condiciones iniciales:

Entrada: Imagen $I(x,y)$ de ceros.
Estado inicial: $\chi(i,j)^0 = I_{ec}(x,y)$, donde $I_{ec}(x,y) = 1 - 2I_e(x,y)^t$.

Para esta red $i=x$, $j=y$. La TCNN discreta está dada por las ecuaciones 3.27 y 3.28, y las plantillas de pesos están dadas por:

$$A = 2 \quad B = 0 \quad Z_c(i,j) = -z_o \qquad (3.29)$$

z_o es una constante que puede ser seleccionada en el intervalo de [-1,1]. De esta manera, la red se modela de la siguiente forma:

$$\chi(n_c+1) = f\left[\chi(n_c)\right] - z_o \qquad (3.30)$$

[17] Disipativo se refiere a que la información de cada celula se extiende al vecindario.

En la ecuación 3.30 z_o es el umbral. Como salida en $y_c(i,j)^{\infty}$ se obtiene una imagen cuya respuesta está dada por las siguientes reglas:

$y_c(x,y)=-1$, si $I_{ec}(x,y) > z_o$.

$y_c(x,y)=1$, si $I_{ec}(x,y) \leq z_o$.

En la Figura 3.14, se puede observar una imagen a escala de grises procesada con la TCNN con $z_o=0.5$ (Figura 3.14b), $z_o=0$ (Figura 3.14c) y $z_o=-0.5$ (Figura 3.14d).

a) b) c) d)

Figura 3.14. Resultado de TCNN **[**116**]**. a). Imagen original. b). Salida con $z_o=0.5$. c). Salida con $z_o=0$. d). Salida con $z_o=-0.5$.

No obstante, como se puede apreciar en el modelo de la ecuación 3.30, la TCNN no realiza una binarización basándose en el vecindario. Por lo tanto, en esta subsección se propone una novedosa forma de binarización basada en la TCNN, la cual se denomina Red Neuronal Celular para Binarización basada en Vecindario (NTCNN). Se trata de una TCNN que define el umbral de un pixel $I_e(x,y)^t$ con base a su nivel de gris y los niveles de grises del vecindario en cada iteración de la red. Para ello, la NTCNN tiene las siguientes plantillas de pesos:

$$A=\begin{bmatrix} a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & a_1^{n_c} & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & 2 & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_1^{n_c} & a_1^{n_c} & a_1^{n_c} & a_2^{n_c} \\ a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} & a_2^{n_c} \end{bmatrix}, a_1<1, a_2<1 \quad B=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad Z=-z_o \tag{3.31}$$

Con estos pesos, se puede definir un modelo similar a la TCNN pero que toma en cuenta información de los vecinos. La red se modela de la siguiente forma:

$$\chi(i,j)^{n_c+1}=2f\left[\chi(i,j)^{n_c}\right]+a_1^{n_c}\sum_{(i_N,j_N)\in N_8} f\left[\chi(i_N,j_N)^{n_c}\right]+a_2^{n_c}\sum_{(i_N,j_N)\in N_{(25-8)}} f\left[\chi(i_N,j_N)^{n_c}\right]-z_o(i,j) \tag{3.32}$$

donde N_8 se refiere a los vecinos 8 del pixel $\rho(x,y)\epsilon I_{ec}(x,y)$, mientras que $N_{(25-8)}$ son los pixeles que envuelven a los pixeles N_8. Esta DTCNN de binarización basada en vecindario

(NTCNN) realiza en las primeras iteraciones una eliminación de ruido y en las siguientes iteraciones, binariza como la TCNN de [116]**.** Las condiciones iniciales de la NTCNN son las mismas que la TCNN, dadas por:

Entrada: Imagen $I(x,y)$ de ceros.
Estado inicial: $\chi(i,j)^0=I_{ec}(x,y)$, donde $I_{ec}(x,y)=1-2I_e(x,y)^t$.

Para esta red $i=x$, $j=y$. En cada iteración n_c la NTCNN le asigna a cada pixel un umbral que depende del parámetro global $z_o(i,j)$ y el producto de los vecinos ponderado por a_1 y a_2, los cuales generan un parámetro que va cayendo en cada iteración ya que si $a_1<1$, $a_1<1$ y $n_c\rightarrow\infty$, entonces $a_1^{n_c}\rightarrow 0$ y $a_2^{n_c}\rightarrow 0$. De esta forma, en cada iteración n_c de la NTCNN, el umbral de cada pixel $Th(i,j)$ está determinado por:

$$Th(i,j)^{n_c}=a_1^{n_c}\sum_{(i_N,j_N)\in N_8} f\left[\chi(i_N,j_N)^{n_c}\right]+a_2^{n_c}\sum_{(i_N,j_N)\in N_{(25-8)}} f\left[\chi(i_N,j_N)^{n_c}\right]-z_o(i,j) \qquad (3.33)$$

Con base a la ecuación 3.32 la salida cuando $n_c\rightarrow\infty$, $y_c(i,j)^{\infty}$ de la NTCNN es una imagen con valores de {-1,1}, donde:

$y_c(i,j)=-1$, si $I_{ec}(x,y) > zc+Nac$.
$y_c(i,j)=1$, si $I_{ec}(x,y) \leq zc+ Nac$.

donde $Nac=a_1^{n_c}\sum_{(i_N,j_N)\in N_8} f\left[\chi(i_N,j_N)^{n_c}\right]+a_2^{n_c}\sum_{(i_N,j_N)\in N_{(25-8)}} f\left[\chi(i_N,j_N)^{n_c}\right]$.

Nac modifica el umbral de cada elemento $y_c(i,j)$ en cada iteración n_c con base a la información del vecindario $y_c(i_N,j_N)$. La NTCNN se realiza durante *NC* iteraciones (en ocho iteraciones termina de realizar la binarización). Luego de las *NC* iteraciones, la salida de la NTCNN es una imagen binarizada que contiene los resultados de la segmentación de objetos dinámicos dada por:

$$Is(x,y)^t=\left[yc(i,j)^8-1\right]/2 \qquad (3.34)$$

La ecuación 3.32 mapea los valores de $Ie(x,y)^t$ al intervalo {-1,1}, de forma que los pixeles cercanos a cero ahora son cercanos a uno y la ecuación 3.34 da una salida $Is(x,y)^t$ en el intervalo {0,1}, y esta contiene el resultado de la segmentación, ya que aquellas regiones de ruido fueron eliminadas, mientras que se conservaron aquellas regiones que representan los objetos dinámicos, aún cuando los niveles de grises en $I_e(x,y)^t$ fueran bajos.

Para ilustrar el funcionamiento de la NTCNN, considérese un pixel $I(x_1,y_1)=1$, $z(x,y)=0$, $a_1=0.3$, $a_2=0$, los vecinos N_8 $I(x_N,y_N)=-1$, entonces la salida en $y(x_1,y_1)^{\infty}=-1$, es decir, durante el proceso iterativo $I(x_1,y_1)=1$ tiende al valor de sus vecinos, como se ve en la Figura 3.15a. Si un pixel $I(x_1,y_1)=-1$, $z_o(x,y)=0$, $a_1=0.3$, $a_2=0$ y los vecinos $I(x_N,y_N)=1$, entonces la salida en $y(x_1,y_1)^{\infty}=1$, es decir, durante las iteraciones, $I(x_1,y_1)=1$ tiende al valor de sus vecinos, lo cual se ve en la Figura 3.15b. Sin embargo, si los valores de los vecinos no son muy diferentes al valor del pixel, entonces la NTCNN se comporta de la misma forma que la TCNN, como en la Figura 3.15c, donde $I(x_1,y_1)=0.3$, $z(x,y)=0$, $a_1=0.1$ y los vecinos $I(x_N,y_N)=-0.3$ (todos los vecinos son -0.3), la salida $y(x_1,y_1)^{\infty}=1$ como se muestra en la Figura 3.15c.

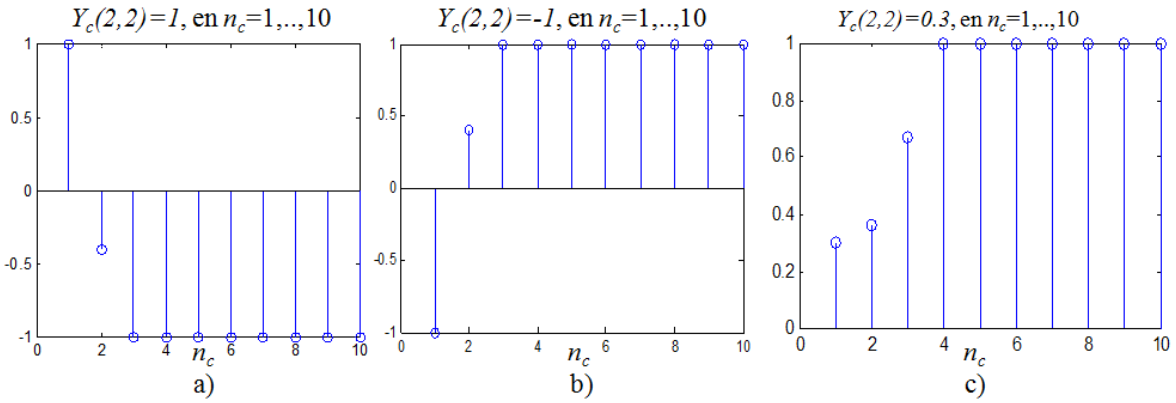


Figura 3.15. Iteraciones de la NTCNN con a). $I(x_1,y_1)=1$, $z(x,y)=0$, $a_1=0.3$, $a_2=0$ y los vecinos $I(x_N,y_N)=-1$. b). $I(x_1,y_1)=-1$, $z(x,y)=0$, $a_1=0.3$, $a_2=0$ y los vecinos $I(x_N,y_N)= 1$.c). $I(x_1,y_1)= 0.3$, $z(x,y)=0$, $a_1=0.1$ y los vecinos $I(x_N,y_N)=-0.3$.

Este modelo de la NTCNN no solo se inspira a partir de la TCNN, también se inspira a partir de la percepción de superficies de la capa cortical de superficies monoculares que se define en la ecuación 2.4, donde a partir de la respuesta difusiva de cada neurona se obtiene un análisis de las superficies dadas por el patrón de entrada.

La NTCNN tiene tres parámetros; a_1, a_2 y z_o, los cuales caracterizan el umbral $Th(i,j)$ en cada pixel $\rho(x,y)\epsilon I_e(x,y)^t$ de forma tal que conforme $a_1\rightarrow 1$ y $a_2\rightarrow 1$, el valor del pixel $y_c(i,j)$ es más dependiente del valor de los pixeles vecinos $y_c(i_N,j_N)$. Si $a_1\rightarrow 0$ y $a_2\rightarrow 0$ el valor de $y_c(i,j)$ es más dependiente del umbral global z_o. Este último parámetro debe tener valores de [-1,1], y si se analiza la ecuación 3.32, si $z_o=1$, todos los pixeles distintos de cero en $I_e(x,y)^t$ serán uno en $Is(x,y)^t$. Si $z_o=-1$, todos los pixeles distintos de uno en $I_e(x,y)^t$ serán cero en $Is(x,y)^t$. Para seleccionar el valor de z_o, se considera que es mejor detectar un objeto

dinámico falso que no poder detectar objetos dinámicos; por lo tanto, basándose en el análisis de $z_o(i,j)$ en la ecuación 3.32, en este trabajo $z_o(i,j)$ se deja constante en $z_o(i,j)=1$. De esta forma, se pueden evitar los problemas de falsos negativos y camuflaje. Para reducir ruido en $Is(x,y)^t$, se utilizan los parámetros de a_1, a_2, ya que con estos parámetros se puede seleccionar para cada pixel un umbral que depende del vecindario. En las siguientes secciones se mostrará como los parámetros a_1, a_2 se ajustan en forma automática.

En la Figura 3.16 se pueden observar dos ejemplos de binarización con NTCNN y uno con umbralización; la Figura 3.16a muestra una binarización con $Th_2 = 0.35$, la cual fue el mejor resultado utilizando este método, mientras que tal como se observa en la Figura 3.16b y 3.16c, la NTCNN genera mejores resultados en la segmentación ya que se reducen los problemas de ruido y a su vez se conserva casi toda la información del objeto dinámico. Como se puede observar en la Figura 3.16b y 3.16c, la NTCNN puede generar salidas con buenos resultados con diferentes valores en a_1 y a_2. En la Figura 3.17 se observa un problema de fondo dinámico crítico (fondo dinámico que cubre todo el cuadro), y como se observa en la Figura 3.17c, la umbralización con $Th_2 = 0.35$ genera mucho ruido, mientras que en el caso de la NTCNN el ruido causado por el fondo dinámico se reduce enormemente, pero conservando el objeto dinámico. En caso de que el umbral Th_2 aumente, el objeto dinámico se pierde.

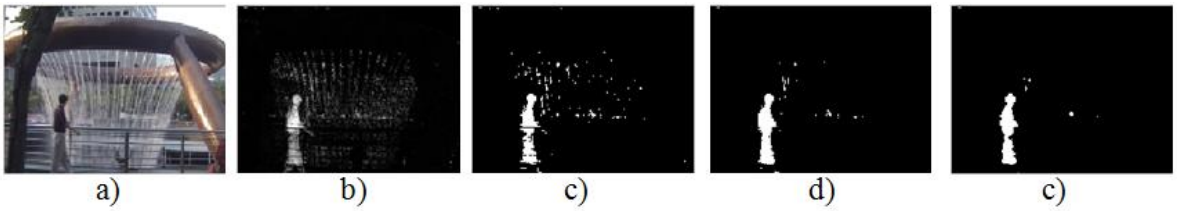


Figura 3.16 Resultados de binarización a) cuadro *t=415* del video '*FT*' b) $Ie(x,y)^t$. c) Umbralización con $Th_2=0.35$. d) $Is(x,y)^t$ con NTCNN y $a_1=0.11$, $a_2=0.11$. e) $Is(x,y)^t$ con NTCNN y $a_1=0.35$, $a_2=0.05$.

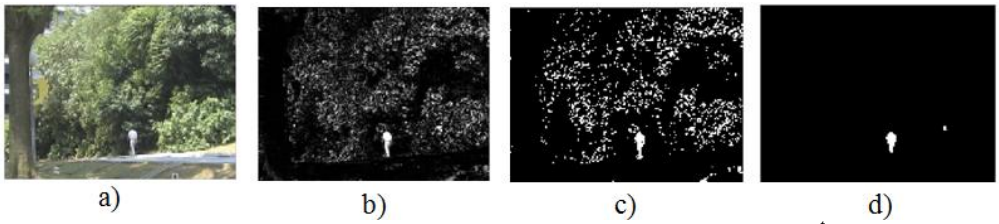


Figura 3.17. Resultados de binarización a) cuadro *t=415* del video '*CAM*' b) $Ie(x,y)^t$. c) Umbralización con $Th_2=0.35$. d) $Is(x,y)^t$ con NTCNN y $a_1=0.55$, $a_2=0.5$.

En conclusión, en el método de segmentación que se va a proponer en las siguientes secciones, la NTCNN tendrá como constantes $NC = 8$, $z_o = 1$ y como parámetros a_1 y a_2.

3.4 Método de segmentación de objetos dinámicos SOM-CNN

En general, la mayor parte de los métodos de segmentación de objetos dinámicos en la literatura se basan en substracción de fondo [104] o flujo óptico [118]. El método de segmentación de objetos dinámicos propuesto en esta tesis se basa en substracción de fondo, ya que es más popular en la literatura y muchas bases de datos utilizadas en la literatura contienen videos que no evalúan la contribución de los métodos de flujo óptico. Como se mencionó anteriormente, DR-SOM es el método de substracción de fondo propuesto en esta tesis y el resultado obtenido de este método es la distancia euclidiana dada por la ecuación 3.26. Este resultado no se puede considerar la segmentación de objetos, ya que DR-SOM obtiene una segmentación suave de movimiento. Para segmentar por completo los objetos dinámicos se utiliza la NTCNN, la cual genera una salida binarizada donde las regiones formadas a partir de conjuntos de pixeles $\rho(x,y) \epsilon Is(x,y)^t$ con valor de uno que se encuentren conectados representan los objetos dinámicos presentes en la escena. Ambos, DR-SOM y NTCNN forman el método de segmentación de objetos dinámicos propuesto el cual se denomina SOM-CNN. SOM se deriva del modelo RESOM que se utiliza para modelar el fondo en DR-SOM y CNN por la NTCNN la red que se utiliza para concluir la segmentación.

Al igual que SOM-CNN, existen múltiples trabajos basados en RNA que realizan segmentación de objetos dinámicos. Como ejemplo están los trabajos basados en SOM como SOBS [13], SSOM **[**14**,**15**,**119] y GSOM [96]. Entre estos métodos, el más recurrente en la literatura es SOBS, el cual se basa en una SOM que representa un patrón de entrada en un mapa de menor dimensión, pero conservando la misma relación topológica **[**13**]**. Este método es muy citado en la literatura por los buenos resultados que reporta, sin embargo, contiene una serie de parámetros que se tienen que ajustar manualmente; es decir, para cada video es necesario seleccionar en los parámetros los valores que permiten una segmentación coherente. SOM Simplificada (SSOM) es un modelo basado en SOM que se utiliza para substracción de fondo, el cual fue propuesto en **[**14**]**. A diferencia de SOBS, SSOM ajusta automáticamente sus parámetros con un módulo basado en lógica difusa propuesto en **[**119**,**15]. *Growing* SOM (GSOM) es un modelo propuesto en [96] donde una SOM jerárquica va analizando la secuencia de video para obtener múltiples modelados de fondo. No obstante, GSOM solo es probado con una secuencia de video. Además de la

SOM existen otros métodos como BNN, el cual se basa en una RNA probabilística cuya arquitectura forma un clasificador bayesiano para definir si un pixel contiene información que pertenece al fondo o es objeto dinámico. Los métodos mencionados anteriormente, se enfocan a resolver problemas de fondos dinámicos. Como estos, existen muchos otros métodos de segmentación de objetos dinámicos basados en RNA, la mayor parte de estos se enfocan a realizar la segmentación de movimiento en videos con problemas específicos. Adicionalmente, existen métodos como SOBS que se pueden utilizar para videos con diferentes problemas en las condiciones del escenario, pero requieren entrenamiento previo o ajuste manual de parámetros.

De acuerdo a la literatura analizada, solo SSOM en [119] es un método diseñado para fondos dinámicos robusto a cambios drásticos en las condiciones del escenario como cambios de iluminación repentinos. El resto de los métodos basados en RNA y modelos probabilísticos diseñados para fondos dinámicos tienen problemas en cambios drásticos en las condiciones del escenario. Sin embargo, una gran ventaja de SOM-CNN es que tiene un conjunto de parámetros que pueden ser utilizados para modelar el aprendizaje de acuerdo a las condiciones del escenario. SOM-CNN se diseñó como un método de segmentación de objetos dinámicos que va a ser robusto a múltiples condiciones en un escenario tales como fondos dinámicos, *bootstraping*, objetos dinámicos estacionarios, ruido, escenarios con bajos contrastes (como escenarios obscuros), cambios graduales y repentinos de iluminación. Por lo tanto, la contribución de SOM-CNN es que es robusto a múltiples problemas en las condiciones del escenario que incluyen desde Objetos dinámicos estacionarios hasta cambios repentinos de iluminación. Además, a diferencia de muchos métodos que son reportados exitosos en varios problemas, SOM-CNN tiene un módulo que ajusta de manera automática sus parámetros. En esta sección 3.4 se va a describir SOM-CNN y la manera en cómo realiza la segmentación.

3.4.1 Arquitectura de SOM-CNN

A grandes rasgos, SOM-CNN se puede ver como un método de modelado de fondo que se complementa con otros dos módulos; un módulo de binarización y otro módulo de ajuste de parámetros. En la Figura 3.18 se muestra la arquitectura general de este modelo, el

cual tiene tres módulos: modelado de fondo basado en DR-SOM, binarización por NTCNN y tercer módulo o de ajuste automático de parámetros.

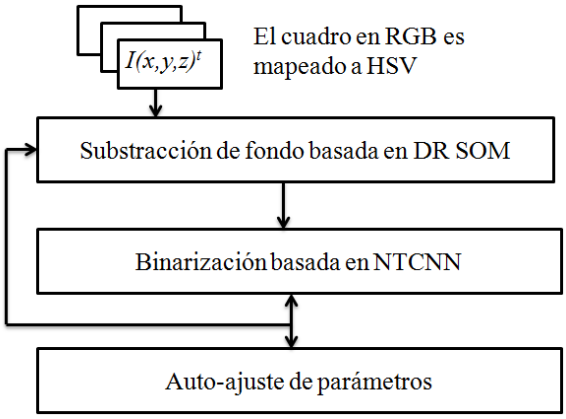


Figura 3.18 Esquema general del método propuesto llamado SOM-CNN.

La entrada de SOM-CNN es un cuadro de la secuencia de video $I(x,y,z)^t$, donde t es el índice de cuadro, (x,y) la posición del pixel y z la componente de color. Aunque los videos están definidos en el espacio RGB ($z=\{R,G,B\}$), SOM-CNN mapea cada cuadro al espacio de color de HSV ($z=\{H,S,V\}$), donde Matiz (H), Saturación (S) y Valor (V) están definidos en **[**120**,**121**,**122**].** Con este espacio de color se mejoran los resultados en videos con fondos dinámicos, además es muy empleado en diversos métodos de segmentación de objetos dinámicos como en **[**13**,**95**,**15**].** Para mejorar el aprendizaje de la RESOM se recurre al manejo en coordenadas cartesianas del espacio de color HSV, de forma que $z=\{X_{hsv},Y_{hsv},Z_{hsv}\}$, donde:

$$X_{hsv} = VS\cos(H) \tag{3.35}$$

$$Y_{hsv} = VSsen(H) \tag{3.36}$$

$$Z_{hsv} = V \tag{3.37}$$

DR-SOM es el primer módulo, pero tal como se muestra en la Figura 3.19, tiene dos modificaciones respecto del modelo original. La primera consiste en que RESOM tiene como entrada los componentes X_{hsv},Y_{hsv},Z_{hsv} de cada cuadro $I(x,y,z)^t$ en vez de los componentes RGB. Una segunda modificación consiste en que RESOM tiene una retroalimentación que viene del módulo de ajuste de parámetros. Dicha retroalimentación se utiliza para que SOM-CNN sintonice los parámetros de aprendizaje de RESOM con base a las condiciones de la secuencia de video.

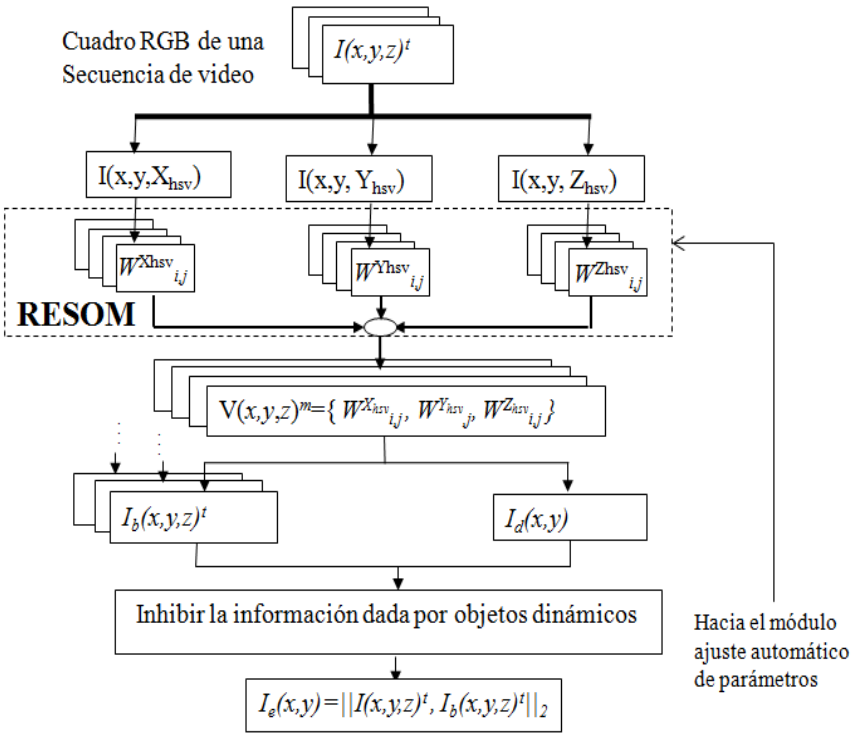
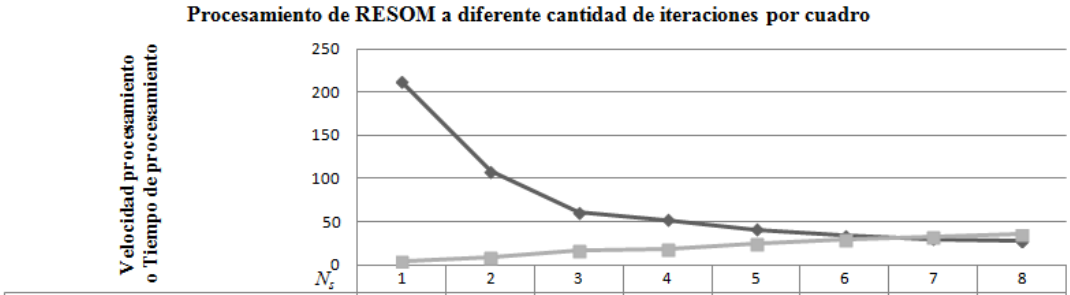


Figura 3.19 Módulo DR-SOM modificado para SOM-CNN.

Los parámetros de DR-SOM son N_s, α_o, σ_s, σ_β, α, $\beta(m)^z$, T_f, t_a, y v_w. Como se vio en la sección 3.2, σ_β, α, $\beta(m)^z$ dependen de α_o, σ_s y T_f (ver ecuaciones 3.14 a 3.16). Al final de la subsección 3.2.2 se mencionó que para un adecuado modelado de fondo basado en RESOM, $N_s = 3$ para $t = 1$ y $N_s = 1$ para $t > 1$. En $t=1$ es necesario que $V_w(x,y,z)^{m,t}$ sea actualizado con todas las neuronas para que en $I_b(x,y,z)^t$ aprenda por completo el primer cuadro como modelado de fondo inicial. Para $t>1$, se actualiza $V_w(x,y,z)^{m,t}$ con una neurona por iteración en cada red para evitar que $I_b(x,y,z)^t$ sea sensible a aprender detalles en cambios sucesivos como ruido. En consecuencia, en SOM-CNN $v_w=M=4$ (se utiliza la ecuación 3.17) en $t=1$ y $v_w=1$ para $t>1$ (se utiliza la ecuación 3.21). Basándose en el comportamiento de estos parámetros, SOM-CNN solo sintoniza α_o, σ_s y T_f para el módulo de DR-SOM, ya que N_s y v_w funcionan como constantes en SOM-CNN y los parámetros, σ_β, α, $\beta(m)^z$ dependen de α_o, σ_s y T_f.

Aunque se puede modelar el aprendizaje por cuadro con N_s, este se dejó constante en t>1 porque afecta directamente en los tiempo de procesamiento, ya que tal como se observa en la gráfica de la Figura 3.20, la velocidad de procesamiento en $N_s=1$ es mucho mayor que $N_s>1$. Por esta razón, este parámetro se dejó constante en $N_s=1$ en SOM-CNN. Con videos

de 120x160, el tiempo de procesamiento de NTCNN en MATLAB® es aproximadamente 4.7 *mseg* y 212 *fps*.



N_s	1	2	3	4	5	6	7	8
Velocidad procesamiento en cuadros por segundo (fps)	212	109	61	52	41	34	30	28
Tiempo Procesamiento en mseg	4.7	9.1	16.2	18.9	24.1	29.1	32.6	34.9

Figura 3.20. Velocidad de procesamiento de la RESOM a diferentes velocidades con videos de 120x160. Los resultados son promedios de velocidades y tiempos de procesamiento de 100 mediciones.

El módulo de Binarización basado en NTCNN tiene cuatro parámetros: *TC*, z_o, a_1 y a_2. Para SOM-CNN *TC=8*, ya que esta es la cantidad necesaria de iteraciones para que NTCNN genere una salida binarizada. Para evitar problemas de falsos negativos, $z_o=1$. Dado que los parámetros *TC* y z_o son constantes en SOM-CNN, los parámetros a_1 y a_2 quedan determinados por el módulo de ajuste de parámetros.

3.4.2 Modulo de Ajuste automático de parámetros

El tercer módulo o módulo de ajuste de parámetros tiene como objetivo manipular el comportamiento de DR-SOM y de NTCNN ajustando en cada cuadro los parámetros de dichas redes. Dicho ajuste se basa en un análisis estadístico de las condiciones del escenario para obtener una segmentación coherente de objetos. Los parámetros que ajusta este módulo son α_o, σ_s y T_f de DR-SOM y a_1, a_2 de NTCNN; DR-SOM modifica el aprendizaje de SOM-CNN y NTCNN puede reducir la cantidad de falsos positivos o falsos negativos en el resultado de la segmentación.

Con base al manejo de estos parámetros, SOM-CNN resuelve problemas de *bootstrapping*, fondos dinámicos, ruido de video, movimiento de cámara, camuflaje, cambios de iluminación graduales y repentinos. Para ello, el tercer módulo de SOM-CNN realiza tres tipos de análisis:

Bootstrapping: Analiza los primeros cuadros para eliminar de $I_b(x,y,z)^t$ posibles objetos dinámicos que se hayan integrado cuando t_a es cero.

Problemas de iluminación: Utilizando operadores estadísticos como Media, Desviación estándar y Entropía del escenario, este análisis verifica si existen cambios de iluminación graduales, repentinos, o las condiciones de contraste en el escenario que propician el camuflaje.

Fondos dinámicos y objetos dinámicos estacionarios: Analiza mediante la cantidad de regiones formadas de la diferencia entre cuadros si puede existir un fondo dinámico. Si es así, se utiliza el valor esperado de la diferencia entre cuadros para analizar si existen objetos dinámicos y/o fondos dinámicos.

Actualización de Parámetros: SOM-CNN asume que el escenario se encuentra en condiciones normales cuando va a iniciar el procesamiento de un video. En estas condiciones no existe ningún problema que cause cambios considerables en el fondo, por lo tanto, se puede tener una capacidad de aprendizaje baja en DR-SOM. Además, en condiciones normales no existe ruido causado por fondos dinámicos, movimiento de cámara, camuflaje, etc. Por lo tanto, se puede tener valores de a_1 y a_2 casi en cero en la NTCNN. Los parámetros dados en condiciones iniciales son $\alpha_s = 0$, $\sigma_s = 0$, $T_f = 35$, $a_1 = 0.05$, $a_2 = 0.01$; T_f define la caída de α a α_o, y σ_β a σ_s en menos de diez cuadros. Los valores de α_s y σ_s no tienen relevancia en los primeros cuadros y más delante se abordará como es que se sintonizan después de la caída definida por T_f. a_1 y a_2 se colocan con valores muy bajos para que todos los pixeles diferentes de cero en $I_e(x,y)^t$ sean uno. No obstante, valores de $a_1 < 0.05$, $a_2 < 0.01$ o $a_1 < 0.06$, $a_2 < 0$ pueden causar problemas de ruido ya que cualquier valor en $I_e(x,y)^t$ entre $[0,\varepsilon]$ puede ser segmentado como parte de objeto dinámico aunque sea un ruido poco significativo.

Después del primer cuadro, los parámetros se sintonizan con base a las condiciones del escenario. Para medir dichas condiciones, se analizan los cambios entre cuadros consecutivos que se obtienen a partir de la diferencia de la información de V de dos cuadros consecutivos. La componente V fue seleccionada ya que por su definición, está muy relacionada con la luminancia presente en cada cuadro [120]. Por lo tanto, para medir los cambios en el escenario, se obtiene el valor esperado de la diferencia entre cuadros dado por:

$$Iv(k)^t = \left| I(k,V)^t - I(k,V)^{t-1} \right| \tag{3.38}$$

$$\mu_v = \frac{1}{K} \sum_k Iv(k)^t \tag{3.39}$$

Como se mencionó al inicio del capítulo, la percepción de movimiento se forma a partir de ciertas características como alerta/atención visual y cambios de luminancia. Para SOM-CNN, la alerta visual es $I_e(x,y)^t$, ya que contiene información de todos los posibles objetos en movimiento. En consecuencia, si se combina la información de alerta visual con cambios de luminancia entonces tendrá mayor plausibilidad con modelos biológicos de percepción de movimiento. Por lo tanto, la sintonización de parámetros en DR-SOM a partir de la información de $Iv(k)^t$ le da a SOM-CNN mecanismos de aprendizaje similares a los modelos perceptuales.

El promedio μ_v mide los cambios de luminancia entre un cuadro y otro, ya que está relacionado a la cantidad de valores diferentes de cero en $Iv(k)^t$ y sus niveles de grises. En la Figura 3.21 se puede observar que si se presenta un objeto dinámico entonces $Iv(x,y)^t$ genera un patrón parecido a los bordes del objeto (Figura 3.21b), si se presenta ruido entonces se generan diversos conjuntos de regiones irregulares como se muestra en la Figura 3.21d. Si los objetos dinámicos son muy reducidos en tamaño en comparación con el escenario o el ruido generado no es muy significativo, $\mu_v \approx 0$. Si llegan a generarse cambios muy fuertes como movimiento de objetos de tamaño considerable respecto a la escena o existe ruido, entonces μ_v tendrá un rango de aproximadamente (0,0.1], como en el ejemplo de la Figura 3.21c, donde μ_v es de 0.00235. No obstante, experimentalmente se pudo observar que incluso en estas condiciones no es necesario aumentar los parámetros de DR-SOM.

Generalmente, μ_v es mayor a 0.1 solo en casos donde existen vibraciones de cámara o fondos dinámicos que son tan severos que las regiones irregulares en $I_e(x,y)^t$ causadas por dicho fondo se juntan formando regiones solidas que cubren la mayor parte del escenario. En estas circunstancias, el ruido será inherente al resultado de la segmentación, por lo que es necesario incrementar el valor de los parámetros de SOM-CNN para reducir el ruido.

a) b) c) d)

Figura 3.21. Resultados de $Iv(x,y)^t$. a) Video 'LB', $t=380$. b). $Iv(x,y)^t$ de a). con $\mu_v=0.00991$. c) Video 'CAM', $t=580$. d). $Iv(x,y)^t$ de c). con $\mu_v=0.0235$.

Por lo tanto, se diseñó una sintonización de los parámetros de aprendizaje basada en la información de μ_v, ya que puede indicar si los cambios en el escenario son lo suficientemente significativos para incrementar los valores de los parámetros α_o, σ_s, a_1, a_2. Esta sintonización se basa en una función sigmoidal dada por:

$$s_d(s_1, s_2, \mu_d) = \frac{s_1}{1+\exp\left(-40(10\mu_d - s_2)\right)} \tag{3.40}$$

La función sigmoidal esta sintonizada para que aumente los valores de los parámetros cuando μ_d tenga valores cercanos a 0.1, s_1 es la amplitud máxima de $s_d(s_1,s_2,\mu_d)$, s_2 es un parámetro que define a partir de qué punto inicia la pendiente de la función sigmoidal y μ_d es un promedio dado por:

$$\mu_d = \frac{1}{\Lambda}\sum_{\tau=t-\Lambda}^{t} \mu_v^{\tau} \tag{3.41}$$

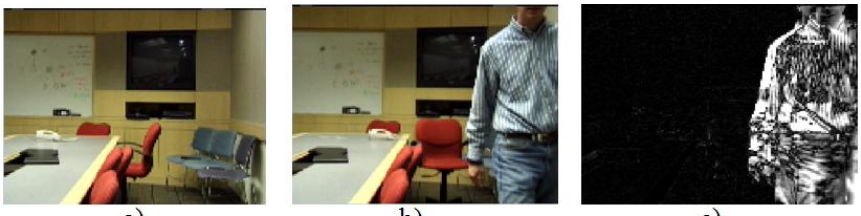
donde Λ es una ventana de tiempo (número de cuadros). Λ permite medir cómo va cambiando el escenario a lo largo de los últimos Λ cuadros. Para SOM-CNN, si $t > 300$ $\Lambda = 300$, si no $\Lambda = t$; cualquier valor puede ser seleccionado para Λ, pero se recomienda sea un número que no permita un aumento en μ_d cuando existan cambios drásticos de μ_v que no representen la dinámica usual del escenario. Con base a la ecuación 3.40, los parámetros α_s, σ_s, a_1, a_2 están dados por:

$$\alpha_s = \alpha_o + s_d(0.3, s_2, \mu_d) \tag{3.42}$$
$$\sigma_s = \sigma_o + s_d(0.5, s_2, \mu_d) \tag{3.43}$$
$$a_1 = a_{1o} + s_d(0.1, s_2, \mu_d) \tag{3.44}$$
$$a_2 = a_{2o} + s_d(0.1, s_2, \mu_d) \tag{3.45}$$

En condiciones normales, $\alpha_o = 0$, $\sigma_o = 0$, $T_f = 35$, $a_{1o} = 0.05$, $a_{2o} = 0.01$ y $s_2 = 0.4$. Dado que en condiciones normales no existe ruido significativo, se propone $\alpha_o = 0$, $\sigma_o = 0$

para que se detecten los objetos dinámicos que están estacionarios la mayor cantidad de cuadros. Con $a_{1o} = 0.05$, $a_{2o} = 0.01$, para evitar falsos negativos pero sin segmentar ruido poco significativo. Estos parámetros cambian su valor dependiendo de las condiciones del escenario (normales, fondo dinámico, escenario obscuro, etc.), lo cual se analizará posteriormente.

Como se acaba de mencionar, se utiliza μ_d en vez de μ_v, para evitar incrementar el valor de los parámetros cuando se presentan cambios abruptos transitorios como un fuerte ruido temporal, objetos dinámicos de gran tamaño que pasan por el escenario por unos cuantos cuadros o basura en el lente de la cámara. Por ejemplo, en el video '*MO*' (Figura 3.22a) tiene como objeto dinámico una persona que entre $t = 1460$ y $t = 1500$ cruza muy cerca de la cámara y debido a los diferentes tonos de la vestimenta, se generan fuertes cambios entre un cuadro y otro como se ve en la Figura 3.22c, por lo que μ_v genera valores hasta 0.12 como se observa en la Figura 3.23a. Si μ_v manejara los parámetros de aprendizaje en la ecuación 3.40, entonces $I_b(x,y,z)^t$ aprendería dicho objeto dinámico causando un "fantasma" en $I_e(x,y)^t$ en cuadros posteriores y por tanto errores en la segmentación. Si se utiliza μ_d en la ecuación 3.40, entonces los parámetros de aprendizaje aumentaran, pero no de forma significativa como se ve en la Figura 3.23a, ya que μ_d representa la dinámica en el escenario en ventanas de tiempo suficientemente grandes para que los cambios abruptos transitorios no sean significativos. En la Figura 3.23b se observa otro ejemplo de los valores de μ_d y μ_v donde en el video 'CAM' μ_d se mantiene estable a pesar de los cambios causados por el Fondo dinámico y los múltiples objetos de distinto tamaño que pasan por el escenario.



a) b) c)

Figura 3.22. Cambios significativos en video 'MO'. a). $I(x,y,z)^1$. b). $I(x,y,z)^{1498}$. c). $Iv(x,y)^{1498}$.

En la Tabla 3.2 se puede observar el promedio $\mu_{\mu d}$ y desviación estándar $d_{\mu d}$ de todos los valores de μ_d a lo largo de secuencias de video completas. Como se mencionó al inicio, los videos '*MO*', '*P2001_1*', '*P2009_L1_1*', '*P2009_L1_5*' tienen condiciones normales,

los videos '*LS*' y '*LB*' tienen cambios de iluminación y el resto de los videos tienen fondos dinámicos. Como se observa en la Tabla 3.2, μ_d no es útil para detectar alguno de los problemas, pero si indica el grado de variación en el escenario; por ejemplo, videos con fondos dinámicos como '*WS*' y '*FT*' tienen valores similares a videos sin este problema como '*LS*' '*P2009_L1_5*' y '*P2001_1*'. Los videos '*LS*' y '*LB*' tienen cambios de iluminación repentinos, pero al ser cambios abruptos, estos no se reflejan en μ_d. Esto se debe a que μ_d indica que tanto varía el escenario debido al movimiento de diferentes objetos tanto de fondo como dinámicos. Posteriormente se mostrará que el valor de s_2 indica el grado en que μ_d va a influir en los parámetros de DR-SOM y NTCNN.

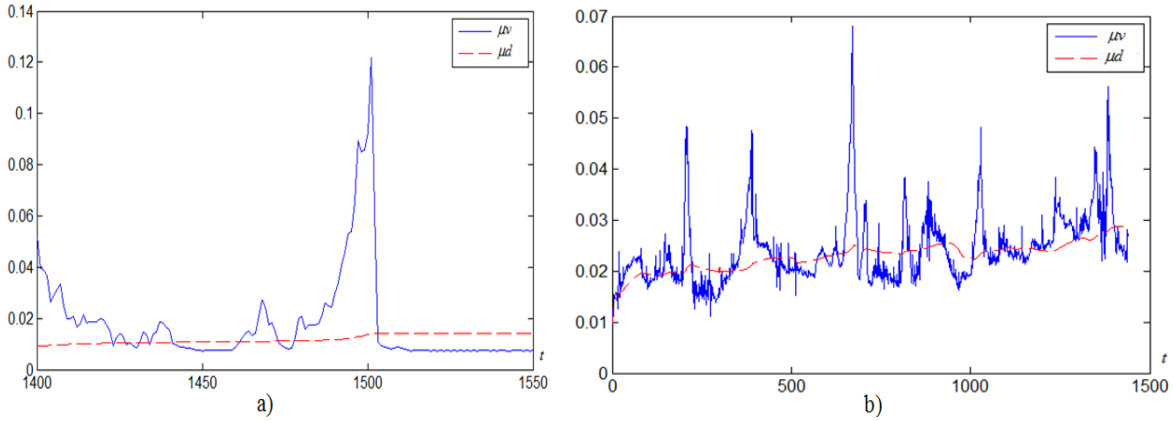


Figura 3.23. Valores de μ_v y μ_d en dos secuencias de video. a). Video 'MO', t=[1400:1550]. b). Video 'CAM'.

Tabla 3.2. Valores de $\mu_{\mu d}$ y $d_{\mu d}$ de varias secuencias de video

Video	**MO**	**CAM**	**LB**	**LS**	**P2001_1**	**SS**	**P2009_L1_1**	**P2009_L1_5**	**MR**	**WS**	**FT**
$\mu_{\mu d}$	0.0097	0.0229	0.0078	0.012	0.0124	0.0311	0.0096	0.0131	0.0076	0.0126	0.0163
$d_{\mu d}$	0.0027	0.0027	0.0006	0.0033	0.00047	0.0036	0.00053	0.0014	0.00074	0.0008	0.00042

Reducción de problemas causados por *Bootstrapping*. En SOM-CNN $I_b(x,y,z)^t$ es el Modelado de fondo a lo largo de toda la secuencia de video. No obstante, si en el primer cuadro del video existen objetos dinámicos, estos se integrarán al fondo ya que de acuerdo a la sección 3.2, $\omega(k,m)^{z,3}=I(k,z)^1$. Si estos objetos dinámicos cambian de posición en los siguientes cuadros, entonces es posible eliminarlos de $I_b(x,y,z)^t$ ya que en cada cuadro se actualiza el fondo. Para ello, RESOM debe aprender los cambios en los primeros cuadros, pero sin que afecte al Modelado de fondo con objetos dinámicos presentes en cuadros posteriores.

Para caracterizar el aprendizaje de RESOM en los primeros cuadros, se puede recurrir al parámetro T_f, el cual de acuerdo a las ecuaciones 3.14 y 3.16 determina la caída de α y σ_β

hasta α_s y σ_s, de forma que T_f es proporcional a la caída de estos parámetros tal como se ve en la Figura 3.24, donde se observa la caída del valor de los parámetros cuando $T_f = 35$ (Figura 3.24a) y $T_f = 100$ (Figura 3.24b). Con base a esto, se puede asumir que T_f puede ayudar a modelar el fondo en caso de que existan objetos dinámicos al inicio de la secuencia, ya que se puede aumentar el valor inicial de T_f para que SOM-CNN los elimine rápidamente de $Ib(x,y,z)^t$. En condiciones iniciales, el valor $T_f = 35$ para que en menos de diez cuadros α y σ_β caigan a α_s y σ_s. Para determinar si existen objetos dinámicos al inicio de la secuencia, SOM-CNN contiene un algoritmo que revisa la media de la diferencia entre el primero y segundo cuadro dada por:

$$I_n(x,y)^2 = I(x,y,V)^2 - I(x,y,V)^1 \tag{3.46}$$

$$\mu_n = \sum_x \sum_y I_n(x,y)^2 \tag{3.47}$$

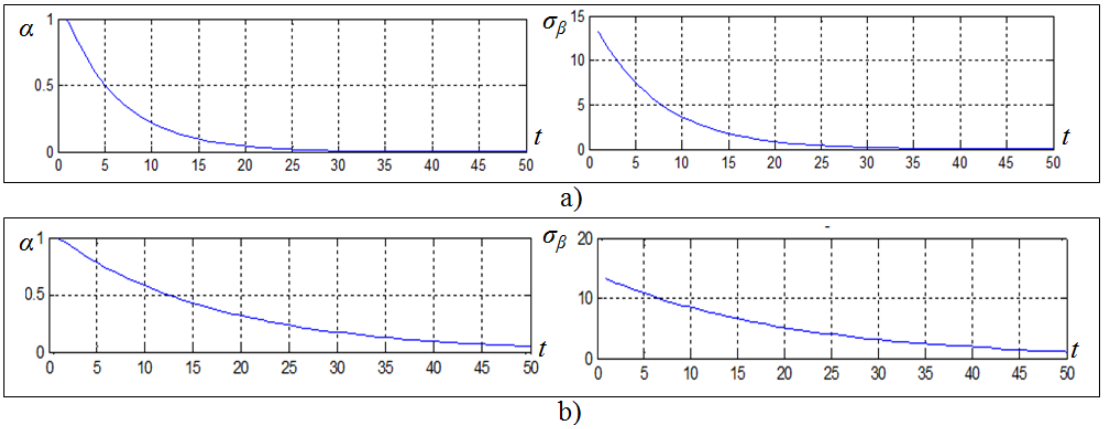


Figura 3.24. Parámetros de aprendizaje en los primeros cuadros de la secuencia de video donde α_s=0.5 y σ_β=0.5 además a) T_f=35. b) T_f=100.

Si μ_n es cero, entonces no hay movimiento, por lo tanto, SOM-CNN asume que no hay objetos dinámicos y T_f se mantiene en $T_f = 35$ para el resto de la secuencia de video. Si μ_n es diferente de cero, SOM-CNN asume que $I_b(x,y,z)^t$ contiene objetos dinámicos por tanto, aumenta el valor de T_f a 100. Al aumentar el valor de T_f, SOM-CNN puede remover "fantasmas" (objetos dinámicos falsos) derivados del primer cuadro de $I_b(x,y,z)^t$ en 15 cuadros (con $T_f = 15$ los "fantasmas" se remueven en más de 50 cuadros), como se ve en la Figura 3.25b y 3.25d donde el resultado de la segmentación de objetos dinámicos no muestra los "fantasmas" derivados del modelado de fondo inicial, (los cuales están en el cuadro inicial mostrado en la Figura 3.1). No obstante, aumentar el valor de T_f también implica que un objeto dinámico que se mueve despacio se integre al fondo lo cual sucede

en la persona rodeada con circulo en la Figura 3.25c, por tanto, T_f se mantiene en 100 solo en los casos que existan objetos dinámicos al inicio de la secuencia, de otra manera se mantiene en 35 para evitar que estos objetos se integren a $I_b(x,y,z)^t$.

Figura 3.25. *Bootstrapping* en dos secuencias de video. a) Video '*P2009_L1_1*', *t=24*. b) $Ie(x,y)^t$ de a). c) Video 'P2009_L1_5', *t=24*. d) $Ie(x,y)^t$ de a).

Análisis de problemas de iluminación y bajos contrastes. Respecto a las condiciones de iluminación, un escenario se puede clasificar en este trabajo de la siguiente forma:

a) Condiciones normales: El video tiene condiciones de iluminación que mantiene con buen contraste las características cromáticas de los objetos en el escenario.

b) Escenarios obscuros y con bajo contraste: Los objetos del escenario tienen condiciones cromáticas que propician problemas de camuflaje y falsos negativos por el bajo contraste de un escenario.

c) Cambios de iluminación graduales y repentinos: Cuando las características cromáticas del escenario cambian, el modelado de fondo debe seguir dicho cambio para no generar problemas de falsos positivos.

En este trabajo se considera que en condiciones normales no existen problemas derivados de la iluminación. No obstante, en condiciones de cambios de iluminación, escenarios obscuros y con bajo contraste SOM-CNN realiza ciertos ajustes para poder mejorar el resultado de la segmentación. Para conocer el estado de la iluminación, se analiza estadísticamente el cuadro actual $I(x,y,z)^t$ con la media, la desviación estándar y la entropía dados por:

$$\mu = \frac{1}{3K}\sum_{z}\sum_{k} I(k,z)^t \tag{3.48}$$

$$\sigma = \sqrt{\frac{1}{3K}\sum_{z}\sum_{k}\left(I(k,z)^t - \mu\right)^2} \tag{3.49}$$

$$E = -\sum_{l} pdf(l)\log_2(pdf(l)) \tag{3.50}$$

donde *pdf(l)* es la función de probabilidad del nivel de gris *l*. De acuerdo a **[**109**,**110], los parámetros de σ y μ son muy utilizadas en diversos métodos para analizar la iluminación. Además, en [109], se argumenta que de acuerdo con la teoría de entropía, se puede medir los cambios de iluminación y se utiliza este parámetro para medir los cambios de iluminación en el método *ISBS* propuesto en [109], obteniéndose los mejores resultados en análisis de cambios de iluminación en la literatura analizada. Basándose en esto, la entropía *E* se utilizó en este trabajo para medir cambios de iluminación graduales y repentinos, y basándose en un análisis del comportamiento de los parámetros σ, μ y *E*, se pudo observar que σ, μ tienen mejor precisión para detectar si un escenario es obscuro o está en condiciones normales.

Escenarios obscuros y de bajos contrastes. Esta categoría se refiere a todos aquellos videos que tienen características cromáticas que propician problemas de camuflaje. Estos videos pueden derivarse de escenarios obscuros o de videos adquiridos con sensores que son sensibles solo a intensidades de grises. En condiciones de obscuridad o escalas de grises, los valores que definen colores e intensidades tienden a ser menores que en condiciones normales cuando se trabaja con espacio de color HSV. De acuerdo a las ecuaciones 3.35 a 3.37, el espacio HSV tiene una forma geométrica similar a un cono (Figura 3.26), donde H es el ángulo que define el giro del cono, SxV es el radio del cono en cada superficie de V. En un escenario obscuro, los valores de V son muy bajos, normalmente menores a 0.2 [123], causando que los colores se mapeen en las partes más bajas del cono, lo cual genera componentes X_{hsv} y Y_{hsv} muy bajos (ecuaciones 3.35 y 3.36). Esto causa que los objetos tengan colores muy similares cuando $z=\{X_{hsv},Y_{hsv},X_{hsv}\}$ en $I(x,y,z)^t$.

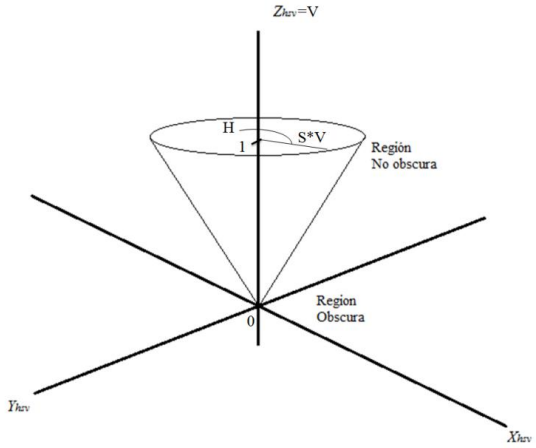


Figura 3.26. Espacio de color HSV.

En el caso de videos en escala de grises, los valores de S y H siempre son cero, causando problemas de camuflaje en objetos con la misma luminancia. En ambos casos, escenarios obscuros y niveles de grises es necesario tener los valores de a_1 y a_2 más bajos que en condiciones normales para reducir falsos negativos, ya que los valores de $I_e(x,y)^t$ causados por objetos dinámicos o ruido son menores que en condiciones normales. No obstante, los parámetros de aprendizaje deben aumentar ligeramente sus valores ya que si el bajo contraste es causado por un monitoreo de noche, puede existir ruido nocturno causado por fuentes de iluminación lejanas que cambian eventualmente de estado. En SOM-CNN, si se detecta que el escenario es obscuro o está a escala de grises, $\alpha_o = 0.15$, $s_o = 0.2$, $a_{1o} = 0.03$, $a_{2o} = 0$. En los parámetros α_o y s_o se determinaron los valores mencionados para evitar errores causados por ruidos nocturnos y en a_{1o}, a_{2o} para reducir los falsos negativos pero evitar que pixeles con niveles entre $[0,\varepsilon]$ causen ruido.

Para ajustar los parámetros, SOM-CNN necesita determinar si existe una condición de escenario obscuro o no existe color. En el caso de escenarios que son solo a escalas de grises es sencillo, ya que el valor esperado μ_H en H y el valor esperado μ_S S debe ser cero. Para el caso de escenarios obscuros también es sencillo, ya que σ, μ se pueden utilizar para analizar condiciones de iluminación. E no se puede utilizar para medir las condiciones de obscuridad, ya que al ser una función logarítmica tiene una tasa de variación que cambia dependiendo de las condiciones de iluminación. μ indica el valor esperado, el cual es proporcional a las condiciones de iluminación si los objetos contenidos en el escenario tienen una iluminación homogéneamente distribuida como en la Figura 3.27a. No obstante, en ocasiones μ tiene valores bajos debido a escenarios que tienen objetos con contrastes muy diferentes, pero dominan los objetos obscuros como en la Figura 3.27b. En escenarios con estas condiciones, σ es más alta que condiciones de obscuridad.

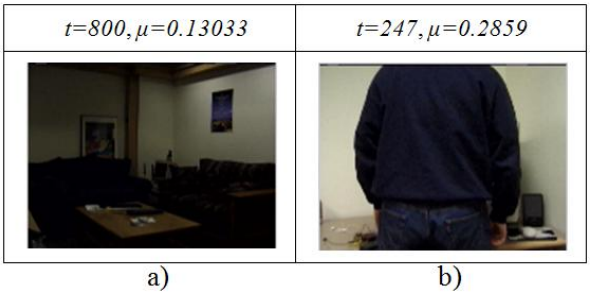


Figura 3.27. Escenarios con media μ baja. a) Escenario con bajas condiciones de iluminación de video '*TD*'. b) Escenario con diferentes condiciones de iluminación de video '*CF*'.

De acuerdo a un análisis, se concluyó que en escenarios en obscuridad que causan camuflaje, $\mu < 0.25$ y $\sigma < 0.24$. En estas condiciones, se generan problemas de falsos negativos, causados por camuflaje y problemas de falsos positivos causados por ruidos nocturnos. Por tanto se cambian los parámetros de aprendizaje como se mencionó anteriormente. Por lo tanto, con base a los valores de α_o, s_o, a_{1o} y a_{2o} en condiciones de obscuridad y bajos contraste se diseñó la siguiente regla:

$$\begin{aligned} &si \quad \left(\mu < 0.25 \wedge \sigma < Th_3\right) \vee \left(\mu_H < \varepsilon \wedge \mu_S < \varepsilon\right) \\ &\quad entonces \quad \alpha_0 = 0.15, \sigma_s = 0.2, a_{10} = 0.03, a_{20} = 0 \end{aligned} \tag{3.51}$$

En la Figura 3.28 se puede observar un ejemplo donde las condiciones de baja iluminación podrían causar problemas de segmentación. El objeto dinámico esta justo después de un cambio de iluminación gradual que dura alrededor de 1800 cuadros, lo cual no es problema para SOM-CNN ya que este cambio lo pudo seguir de forma satisfactoria en $I_b(x,y,z)^t$.

a) b) c)

Figura 3.28. Ejemplo de segmentación en escenario obscuro y con cambio de iluminación gradual. a) $I(x,y,z)^t$. b) $Id(x,y)^t$. c) Ground truth[18].

Cambios de iluminación. Como se ha mencionado anteriormente, el método más exitoso para analizar cambios de iluminación es ISBS [109], donde el análisis de *E* se basa en la diferencia de entropía entre cuadros consecutivos. Aunque esta diferencia es útil en condiciones de cambios de iluminación repentina, no lo es para condiciones de cambios de iluminación gradual. En la Figura 3.29 se puede observar las señales de *E* de cuatro videos; los videos de '*CF*' y '*WT*' contienen Fondos dinámicos y objetos dinámicos de tamaño significativo, el caso de '*TD*' muestra un cambio de iluminación gradual y el caso de '*CM*' tiene un cambio de iluminación repentino. En estos videos como en otros, existe un comportamiento muy similar, donde la entropía cambia abruptamente al presentarse cambios de iluminación, en cambios de iluminación gradual la entropía cambia

[18]Ground truth se refiere a una imagen que contiene la información correcta que el método de segmentación debe obtener en un determinado cuadro.

gradualmente, en Fondos dinámicos la entropía se mantiene sin cambios y cambia muy poco al presentarse objetos dinámicos que cubren la mayor parte del escenario.

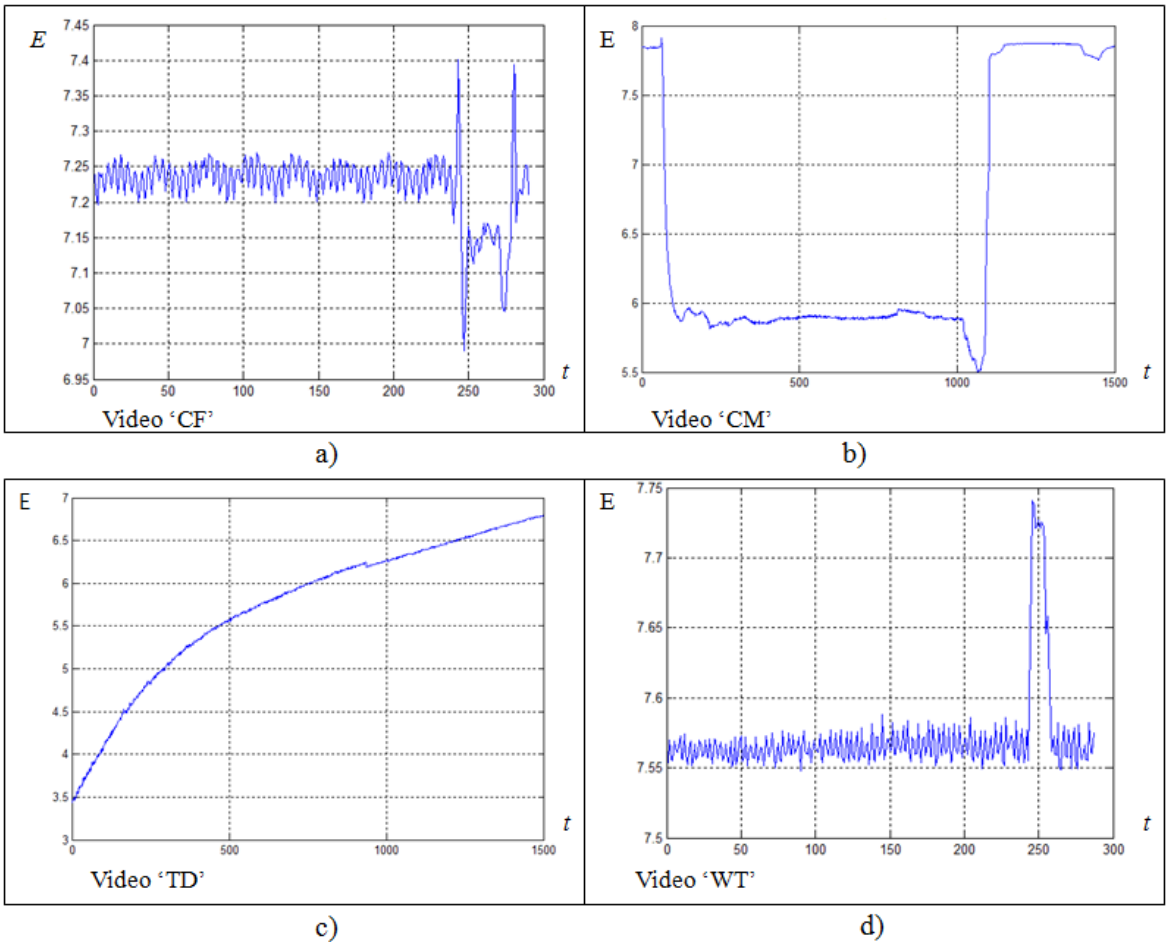


Figura 3.29. Valores de Entropía en diferentes videos. a) Video 'CF' con objeto dinámico de gran tamaño respecto al escenario. b) Video 'CM' con cambios de iluminación repentinos. c) Video 'DT' con cambio de iluminación gradual. d) Video 'WT' con fondo dinámico.

Si en un escenario se genera un cambio de iluminación que empieza a causar falsos positivos en $Is(x,y)^t$, existe una diferencia de entropía significativa. Cuando existen condiciones normales no hay cambios en E aún cuando existan objetos dinámicos. En condiciones de obscuridad habrá cambios más significativos en la entropía que en condiciones normales. Estas situaciones descritas se deben al comportamiento logarítmico de E. Es decir, si ΔE representa un cambio de E y ΔL un cambio de luminancia entre un cuadro y otro, entonces en condiciones normales se puede tener una constante C_1 dada por $C_1 = \Delta E_1 / \Delta L$ y en condiciones de obscuridad una constante C_2 dada por $C_2 = \Delta E_2 / \Delta L$. De tal manera que $C_2 > C_1$ y $|C_1 - C_2|$ será mayor que cualquier cambio de entropía causado por

el tránsito de objetos en un escenario con luminancia casi constante. Por tanto, para detectar cambios de iluminación graduales o repentinos, se recurrió a la siguiente regla:

$$t_a = \begin{cases} 0 & \left|E^{t_b} - E^t\right| > DE \\ t_a + 1 & otra \quad forma \end{cases} \tag{3.52}$$

En condiciones iniciales, $t_b=1$, luego cuando $| E^{t_b} - E^t | > DE$, se actualiza t_b a $t_b=t$ para evaluar la diferencia entre esa nueva entropía y la entropía de los siguientes cuadros. De esta manera, los parámetros de aprendizaje α y σ_β se reinician (ver ecuaciones 3.14 y 3.16), causando que SOM-CNN aprenda las nuevas condiciones del escenario en el cuadro después que $| E^{t_b} - E^t | > DE$, es decir cuando el cambio de iluminación sea significativo respecto al cuadro t_b. El valor *DE* depende de las condiciones de iluminación, ya que los valores de ΔE son muy distintos que en condiciones de obscuridad o condiciones normales. En condiciones normales, existen cambios de iluminación que se pueden definir desde $DE=0.43$. Este valor de *DE* no varía mucho hasta que las condiciones del escenario son muy obscuras lo cual sucede cuando $\sigma<0.15$ y μ tenga valores cercanos a cero; en estas condiciones, *E* tiene cambia de forma muy significativa aún cuando solo se presente ruido y cuando un objeto dinámico cruza por el escenario en estas condiciones, $DE < 3$. Por tanto, en estas condiciones, $DE = 3$. Un aspecto interesante del sistema visual humano es que tiene una respuesta logarítmica respecto al brillo [124] al igual que la entropía.

Aunado a esto, un cambio repentino de iluminación no se genera de un cuadro a otro en una secuencia de video, puede tomar varios cuadros (a veces hasta 20 cuadros); causando que t_b se actualice constantemente porque se generan varios cambios de entropía mayores a *DE* en estos cuadros de transición. Esto puede hacer que $I_b(x,y,z)^t$ acumule información de estos cambios, causando regiones de falsos positivos. Con el objetivo de eliminar estas regiones de falsos positivos, se utiliza la información de $Iv(x,y)^t$ para indicar cuáles son las regiones realmente pertenecen a objetos dinámicos y cuales regiones solo son información acumulada en $I_b(x,y,z)^t$. Esto es, mientras $t_b < 15$ (cantidad de cuadros que dura la condición de *bootstrapping*) pueden existir regiones fantasmas en $I_b(x,y,z)^t$ y para eliminarlas SOM-CNN realiza el siguiente algoritmo:

1. Mediante conectividad 8, realizar un etiquetado en $Is(x,y)^t$ con *LBmax* regiones R_g.

2. Se define un vector de banderas *FlagLabel*(*id*), donde *id=1,..., LBmax* con cada bandera inicializada en cero. Cada elemento del vector representa una región en $I_{d8}(x,y)$ y esta bandera indica si se habilita o deshabilita dicha región si se puede correlacionar con $Iv(x,y)^t$. Es decir:

$$FlagLabel(id)=\begin{cases}1 & if \quad \left[\left(I_{d8}(x_1,y_1)\,\&\,Iv(x_1,y_1)\right)=1 \quad \wedge \quad (x_1,y_1)\in R_g\right] \\ 0 & otherwise\end{cases} \tag{3.53}$$

donde (x_1,y_1) es el índice de un pixel $\rho(x,y)\epsilon R\ \epsilon I_{d8}(x,y)^t$.

3. Definir una imagen con ceros $I_s(x,y)=0$. Luego, a cada elemento de $I_s(x,y)$ se le asignará valor de uno si se cumple la siguiente condición:

$$I_s(x_1,y_1)=\begin{cases}1 & if \quad \left[FlagLabel(id)>0 \quad \wedge \quad (x_1,y_1)\in R_g\right] \\ 0 & Otherwise\end{cases} \tag{3.54}$$

4. El resultado es un cuadro $I_s(x,y)^t$ que contiene las regiones que representan los objetos que coinciden con patrones de movimiento. Estas regiones, son las que se consideran objetos dinámicos. Luego se reasigna $Is(x,y)^t=I_s(x,y)^t$ para obtener una segmentación de objetos dinámicos.

En la Figura 3.30 se puede observar un ejemplo de este algoritmo; en *t = 1855* del video '*LS*' inicia una transición obscuridad-luz que contiene varios cambios de Entropía entre un cuadro y otro de 0.6, de esta forma en *t = 1866* el resultado de la segmentación aún contiene regiones no correspondientes al objeto dinámico. No obstante, en *t = 1866* ya no se presenta un cambio de iluminación, de manera que $Iv(x,y)^t$ contiene información solo del objeto dinámico, de forma que se puede utilizar esta información para retirar el ruido y se puede obtener el objeto dinámico.

Figura 3.30. Segmentación de objetos dinámicos en cambios de iluminación. a) Video '*LS*' *t=1866*. b) $I_v(x,y)^t$. c) $Id(x,y)^t$ antes de la reducción de reigones. d) $Is(x,y)^t$ dedspués de la reducción de regiones. e) Ground truth.

En la Figura 3.31 se puede observar otro ejemplo del video '*CM*', antes de un cambio de iluminación (Figura 3.31a) y después de un cambio de iluminación (Figura 3.31d). Estas

figuras están antes y después de una transición obscuridad-luz. El caso de la Figura 3.31a se puede observar un escenario obscuro que causa problemas de camuflaje en la segmentación del objeto dinámico. Mientras que en el caso de la Figura 3.31d muestra el mismo escenario poco después de haberse realizado una transición obscuridad-luz, en el que se muestra que el objeto dinámico es segmentado con un problema de ruido derivado de que la persona (objeto dinámico) se está moviendo a muy baja velocidad. No obstante, sin el módulo de cambios de iluminación los resultados en condiciones de obscuridad en el video '*CM*' serian nulos, y en el caso del cuadro *t = 1140* de este video aún no se habría podido adaptar el método de segmentación a las nuevas condiciones de iluminación.

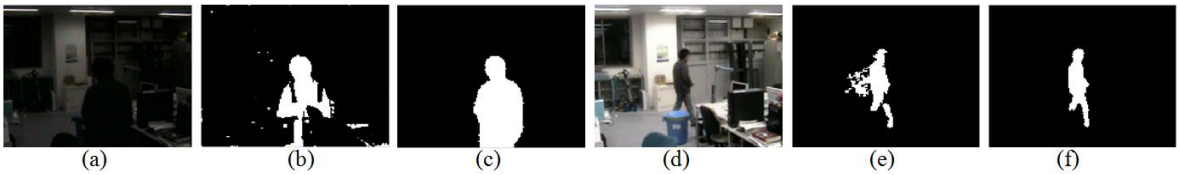


Figura 3.31. Segmentación de objetos dinámicos en cambios de iluminación. a) Video 'CM' t=1050. b) $Is(x,y)^t$. c) Ground truth. d) Video 'CM' t=1140. e) $Is(x,y)^t$. f) Ground truth.

Fondos dinámicos. Un problema ampliamente estudiado en la literatura es el análisis de video con ruido causado por fondos dinámicos **[**11**,**13**,**15**,**96**,**7**,**11**,**10**,**108]. Una manera de reducir este ruido es aumentando la capacidad de aprendizaje en el modelado de fondo para que pueda integrar los cambios de los objetos de fondo que tienen cierto movimiento como árboles, monitores encendidos, fuentes, etc. No obstante, al aumentar la capacidad de aprendizaje los objetos dinámicos estacionarios se pueden integrar también al fondo. Por tanto, en SOM-CNN se manejan los parámetros de aprendizaje de la RESOM para aumentar ligeramente el aprendizaje de manera que se pueda reducir el ruido por fondos dinámicos, pero sin integrar los objetos estacionarios a $I_b(x,y,z)^t$. Para eliminar el ruido restante que no se pudo remover aumentando la capacidad de aprendizaje, SOM-CNN utiliza los parámetros de binarización de la NTCNN. Con base a esto, para solucionar los problemas de Fondos dinámicos sin integrar al fondo objetos dinámicos, SOM-CNN ajusta los parámetros de α, σ_β, a_1 y a_2.

Para manejar los parámetros en forma automática es necesario definir de alguna manera si existe o no un fondo dinámico en la secuencia de video. En un inicio, se esperaba que un análisis estadístico de los niveles de grises indicara si existe o no un fondo dinámico, ya que un fondo dinámico causa una gran cantidad de regiones irregulares en

$I_v(x,y)^t$ como se observa en la Figura 3.32, mientras que un objeto dinámico genera un conjunto de regiones similares a bordes más sólidas y de mayor área que en el caso del fondo dinámico. Sin embargo, dicho análisis estadístico no aporta mucho ya que la cantidad de regiones formadas no tiene relación con los niveles de grises en $I_v(x,y)^t$. La forma más adecuada para determinar si existe o no un fondo dinámico en un escenario es encontrando la cantidad de regiones en $Iv(x,y)^t$, por la gran cantidad de regiones de área muy reducida generada por los Fondos dinámicos como se ve en los videos '*CAM*' y '*WS*' en la Figura 3.32, mientras que los objetos dinámicos sin importar su tamaño generan pocas regiones como se observa en los videos '*LB*' y '*CF*' en la Figura 3.32.

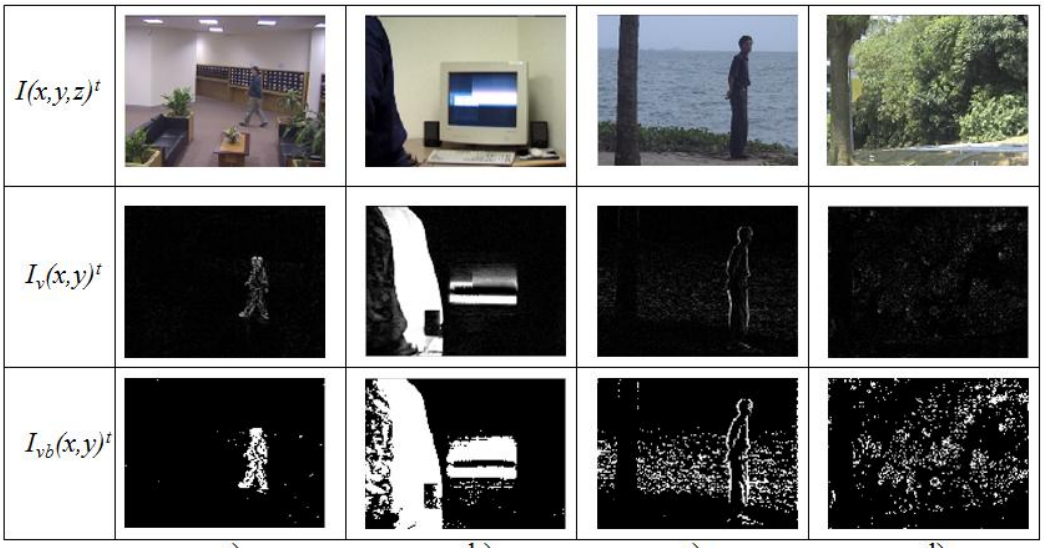


Figura 3.32. Fondos y objetos dinámicos en $I_v(x,y)t$ e $I_{vb}(x,y)t$. a). Video '*LB*', donde $L_v=29$. b). Video '*CF*', donde $L_v=93$. c). Video '*WS*', donde $L_v=392$. d). Video '*CAM*', donde $L_v=519$.

En SOM-CNN el criterio para determinar la cantidad de regiones en $I_v(x,y)^t$ es contar aquellos pixeles con valores mayores a ε. Para ello, SOM-CNN primero genera un cuadro $I_{vb}(x,y)^t$ el cual es una versión binarizada de $Iv(x,y)^t$ con un umbral $Th_4=\varepsilon$ obteniéndose resultados como los de la Figura 3.32. Luego, mediante conectividad 8 se generan las regiones en un cuadro $I_{8b}(x,y)^t$ y se cuenta la cantidad L_v de regiones definidas en $I_{8b}(x,y)^t$. L_v representa la cantidad de regiones en $Iv(x,y)^t$, y como se mencionó previamente, este parámetro se puede utilizar para analizar si existe un Fondo dinámico. Sin embargo, al igual que μ_v, se pueden generar valores en L_v muy altos cuando se generan cambios transitorios. Por tanto, para mantener una medición estable de la cantidad de regiones que el video contiene en una sucesión de cuadros, se utiliza lo siguiente:

$$L_d = \frac{1}{\Lambda} \sum_{\tau=t-\Lambda}^{t} L_v^{\tau} \tag{3.55}$$

Con la ecuación 3.55 se puede obtener una medición de la cantidad de regiones que no varían cuando se presentan cambios abruptos que no representen la dinámica de los objetos en el escenario como se observa en la Figura 3.33.

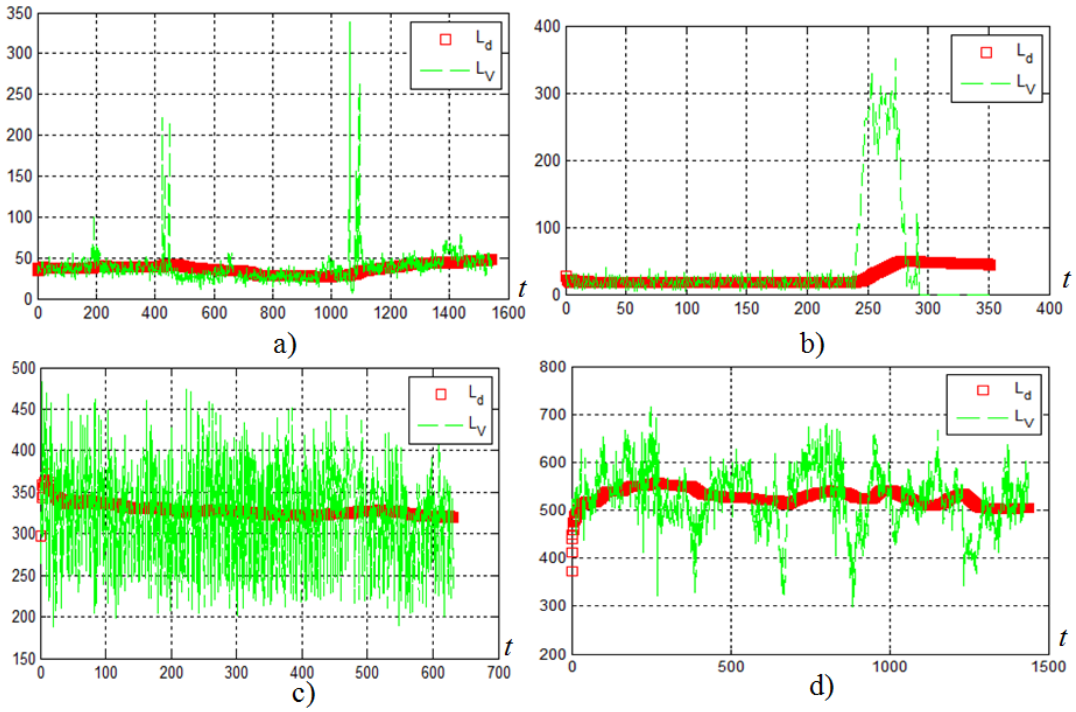


Figura 3.33. Valores de L_v y L_d durante todos los cuadros de: a). Video '*LB*' el promedio de todos los valores de L_d=*37.56*. b). Video '*CF*' el promedio de todos los valores de L_d=*40.48*. c). Video '*WS*' el promedio de todos los valores de L_d=*323.42*. d). Video '*CAM*' el promedio de todos los valores de L_d=*525.33*.

Durante la experimentación con los videos mostrados en la sección 3.1 se realizaron las siguientes conclusiones:

1). Los fondos dinámicos que cubren todo el escenario tienen valores de $L_d > 100$.

2). Existen videos que contienen fondos dinámicos que cubren una porción poco significativa de la escena. Si este fondo dinámico es suficientemente significativo para causar ruido, L_d generalmente es mayor a 50.

3). Los videos en condiciones normales tienen valores de L_d muy bajos. No obstante, en ocasiones existen múltiples objetos dinámicos que pasan por el escenario durante una gran cantidad de cuadros, aumentando los valores de L_d.

4). En todos los videos en condiciones normales, L_d siempre fue menor de 50. Por lo tanto, si $L_d > 50$ entonces no se tienen condiciones normales y existe la posibilidad de que exista un fondo dinámico.

De acuerdo a los puntos mencionados anteriormente, se puede concluir que si $L_d > 50$ entonces es muy posible que exista un fondo dinámico que puede causar errores en la segmentación. Si es así, SOM-CNN ajusta los valores de α_s, σ_s, a_1 y a_2 para que pueda reducir el ruido causado por Fondos dinámicos pero sin afectar la coherencia en el resultado en segmentación de objetos dinámicos. Por lo tanto, SOM-CNN realiza lo siguiente:

$$si \quad L_d > 50 \quad entonces \quad \alpha_o = 0.3, \sigma_o = 0.6, a_1 = 0.08, a_2 = 0.02, s_2 = 0.25 \qquad (3.56)$$

Los valores de α_o y σ_o fueron seleccionados con base a un análisis de diferentes videos con fondos dinámicos para actualizar el fondo lo más rápido posible sin afectar la segmentación de objetos dinámicos estacionarios. Los valores de a_{10} y a_{20} fueron seleccionados con base para segmentar los objetos dinámicos pero eliminar el ruido de fondos dinámicos que no son severos. En consecuencia, con los valores de α_o, σ_o, a_{10} y a_{20}, que se muestran en la ecuación 3.56, SOM-CNN puede conservar la coherencia en la segmentación de objetos dinámicos y reducir el ruido causado por objetos dinámicos. El valor de s_2 tiene como objetivo hacer más sensibles a los parámetros de α_s, σ_s, a_1 y a_2 a μ_d. Esta sensibilidad es necesaria ya que conforme es más severo el fondo dinámico, es necesario aumentar los valores de α_s, σ_s, a_1 y a_2. Es decir, si $\alpha_s \approx \alpha_o$, $\sigma_s \approx \sigma_o$, $a_1 \approx a_{10}$ y $a_2 \approx a_{20}$, SOM-CNN es robusto a escenarios con fondos dinámicos no severos como los presentes en videos como '*WS*' o '*FT*', además puede mantener coherencia en la segmentación de objetos dinámicos. No obstante, conforme el Fondo dinámico se vuelve más severo como en el video '*CAM*', es más complicado contener el ruido presente en $I_e(x,y)^t$, por lo que es necesario aumentar los valores de α_s, σ_s, a_1 y a_2. Para aumentar el valor de estos parámetros se disminuye el valor de s_2 a 0.25 para que la sigmoidal aumente su valor al presentarse ruido en todo el escenario antes que $\mu_d > 0.1$.

En la Figura 3.34 se pueden ver algunos resultados de la segmentación de videos con fondos dinámicos y en la Figura 3.35 se pueden observar resultados en condiciones normales. En todos los casos mostrados, la segmentación fue coherente ya que si se compara $Is(x,y)^t$ con el *Ground truth*, los objetos dinámicos conservan su forma. En la

Figura 3.34 se observa que en ambos videos el ruido se redujo considerablemente y la segmentación de los objetos fue exitosa. Incluso en el video '*WT*' donde el árbol causa valores de μ_d que aumentan de forma significativa a_1 y a_2, el resultado fue exitoso, ya que se logró reducir el ruido y el objeto dinámico fue segmentado conservando su forma.

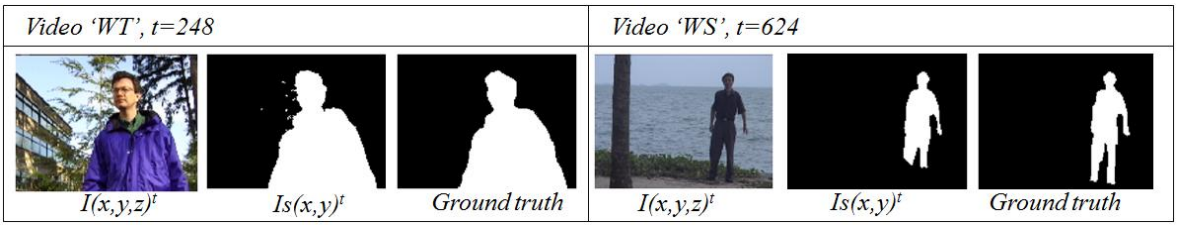


Figura 3.34. Resultados de segmentación dinámica en fondos dinámicos.

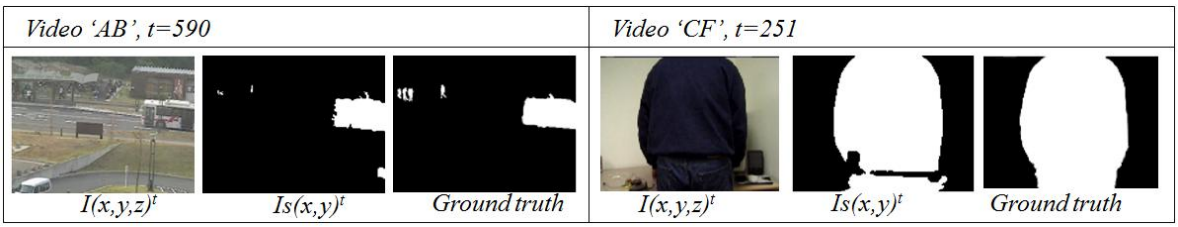


Figura 3.35. Resultados de segmentación dinámica en condiciones normales.

En la Figura 3.35 se observa que los resultados de segmentación en condiciones normales es robusto ya que aunque los valores de μ_d para '*AB*' y '*CF*' es similar a los valores de μ_d para '*WT*' y '*WS*', SOM-CNN puede reconocer con L_d que '*AB*' y '*CF*' no contienen fondos dinámicos.

Si existen condiciones normales, los parámetros se mantienen en sus condiciones iniciales, de manera que $\alpha_o = 0$, $\sigma_o = 0$, $a_{1o} = 0.05$, $a_{2o} = 0.01$ y $s_2 = 0.4$. No obstante, en caso de que existan sombras y reflejos, estos parámetros deben cambiar sus valores para reducir falsos positivos. Tanto en sombras como en reflejos, la entropía cambia de forma poco significativa como los objetos dinámicos, sin embargo, a diferencia de los objetos dinámicos no existen cambios en $I_v(x,y)^t$. Para definir si existen cambios de entropía significativos se diseño la siguiente ventana de entropía:

$$E_d = \frac{1}{\Lambda}\sum_{\tau=t-\Lambda}^{t} E_v^{\tau} \tag{3.57}$$

$$E_p = \left|E_d - E\right| \tag{3.58}$$

La ecuación 3.57 representa la entropía promedio en Λ cuadros y la ecuación 3.58 representa la diferencia entre la entropía actual y la entropía promedio. Cuando se presenta

una sombra o un reflejo, $E_p > 0$ y el valor esperado en $I_v(x,y)^t$ no cambia. Si es así, entonces los parámetros α_o, σ_o, a_{1o} y a_{2o} cambian. α_o, σ_o aumentan para que $I_b(x,y,z)^t$ aprendan los reflejos y sombras que se anexan en el escenario. Los parámetros a_{1o} y a_{2o} aumentan para reducir los falsos positivos causados por sombras y reflejos. Por tanto, se diseño la siguiente ecuación:

$$si \quad \left(E_p < 0.25 \wedge \mu_d = 0\right) \quad entonces \quad \alpha_0 = 0.2, \sigma_s = 0.5, a_{10} = 0.1, a_{20} = 0.1 \qquad (3.59)$$

Los valores propuestos en la ecuación 3.59, fueron seleccionados con base a un análisis para que SOM-CNN se adaptara a sombras y reflejos que se presenten en el escenario.

En resumen, el tercer módulo de SOM-CNN realiza un ajuste de los parámetros T_f, α_s, σ_s, a_1 y a_2 con base a las condiciones del escenario. T_f es seleccionado con base a las condiciones de *bootstrapping*. Los parámetros α_s, σ_s, a_1 y a_2 se adaptan con base a L_d, μ_d y los parámetros auxiliares α_o, σ_o, a_{1o}, a_{2o} y s_2. En la Tabla 3.3 se observa un resumen de los cambios de estos parámetros en las diferentes condiciones ya mencionadas.

Tabla 3.3 Valores de los parámetros auxiliares que manejan α_s, σ_s, a_1 y a_2.

	α_o	σ_o	a_{10}	a_{20}	s_2
Condiciones Normales	0	0	0.05	0.01	0.4
Cond. Normales con reflejos o sombras	0.2	0.5	0.1	0.1	0.4
Fondos dinámicos	0.3	0.6	0.08	0.02	0.25
Escenarios de bajos contrastes	0.15	0.2	0.03	0	0.4

3.4.3 Desempeño del método SOM-CNN respecto a otros métodos. En esta subsección se presenta un análisis comparativo de los resultados de SOM-CNN y otros métodos reportados en la literatura. Para ello, la comparación, se utilizaron los videos de '*LB*', '*CAM*', '*WS*', '*FT*', '*MR*' y '*SS*' ya que en la literatura analizada estos se utilizan comúnmente para comparación de métodos y tienen varios *Ground truth*. Además, se presenta un breve análisis cuantitativo de los videos de la subsección 3.4.2. Los algoritmos fueron probados en una *Workstation Dell Precision* T7500 con procesador Intel *Xeon* y se utilizó MATLAB 2011 para probar los algoritmos. El método SOM-CNN fue probado con videos de tamaños 120x160 y 128x160 y tuvo una velocidad de procesamiento (*frame rate*) promedio de 52 cuadros por segundo (fps), en los cuales el *frame rate* más bajo fue de 47

fps mientras que el más alto fue 58 fps. Con base a este desempeño se puede notar que SOM-CNN es plausible para aplicaciones de tiempo real.

Métricas para la evaluación. Para obtener mediciones del desempeño de SOM-CNN se realizó una comparación de este con otros métodos exitosos de la literatura. Los métodos utilizados para las comparaciones en videos con situaciones de Fondos dinámicos son SSOM [15], CNN [98], SOBS [13], BNN [95] y Li *et al* [125]. Estos métodos fueron seleccionados ya que son comparados y evaluados con los videos '*LB*', '*CAM*', '*WS*', '*FT*', '*MR*' y '*SS*'. Dichos videos miden el desempeño de los algoritmos en escenarios con Fondos dinámicos y cambios de iluminación. Para medir el desempeño, se recurrió a un conjunto de métricas muy utilizadas en la literatura de métodos de segmentación de objetos dinámicos, las cuales son *Presicion* (*P*), *Recall* (*Rc*), *F Measure* (*F1*) y *Similarity* (*Si*) **[**13**,**96**,**109**].** Estas métricas son muy utilizadas en recuperación de información para comparar los resultados obtenidos por un método entre la información que debe obtener. Para definirlas, hay que suponer que un método da como salida cierta información dada por un conjunto de datos y se conoce la información que se debería obtener. La información que se va a evaluar se le denomina de interés, entonces se define lo siguiente:

Verdadero positivo (*tp*): Cantidad de datos de interés diagnosticados correctamente.
Verdadero negativo (*tn*): Cantidad de datos de no interés diagnosticados correctamente.
Falso positivo (*fp*): Cantidad de datos de interés diagnosticados incorrectamente.
Falso negativo (*tn*): Cantidad de datos de no interés diagnosticados incorrectamente.

Con base a esto, *Presicion* (*P*) es un cociente dado por:

$$P = \frac{tp}{tp + fp} \tag{3.60}$$

Este cociente describe un porcentaje de la cantidad de datos de interés diagnosticados correctamente entre todos los datos que el método arroja como positivos. *Recall* (*Rc*) es un cociente dado por:

$$Rc = \frac{tp}{tp + fn} \tag{3.61}$$

Este cociente da un porcentaje de los datos de interés diagnosticado correctamente entre los datos que deben ser positivos. *F measure* relaciona ambos, *P* y *Rc* utilizando una técnica de media armónica ponderada, la cual está dada por:

$$F1 = \frac{2 \cdot Rc \cdot P}{Rc + P} \tag{3.62}$$

Esta medida representa el valor esperado de *Rc* y *P* pero a diferencia de la media normal, no se ve afectada cuando *Rc* o *P* arrojan algún dato crítico cercano a uno o cero. Otra medición es *Similarity* (*Si*), y está dada por:

$$Si = \frac{tp}{tp + fn + fp} \tag{3.63}$$

Esta medida tiene un papel similar a F1, ya que relaciona las medidas de *Rc* y *P* con una medición de comparación de los datos correctamente diagnosticados.

SOM-CNN y otros métodos de segmentación de objetos dinámicos para Fondos dinámicos. En la Tabla 3.4 se puede observar la comparación de SOM-CNN con otros métodos que fueron diseñados específicamente para solucionar problemas de Fondos dinámicos. De estos métodos, SSOM y CNN autoajustan sus parámetros como SOM-CNN, el resto de los métodos necesitan entrenamiento previo o ajustan manualmente sus parámetros.

Tabla 3.4 Resultados promedio de Similaridad (*Si*) con diferentes videos.

	SS	***FT***	***MR***	***LB***	***WS***	***CAM***
SOM-CNN	**0.6124**	0.5096	0.7477	**0.7318**	0.7861	0.4616
SSOM	0.4593	0.5097	0.5581	0.1336	0.7596	0.0988
CNN	0.55	0.6023	0.5074	0.2115	0.7249	0.0566
SOBS	0.577	**0.6554**	**0.8178**	0.6489	**0.8247**	**0.696**
BNN	0.4728	0.4636	0.7368	0.6276	0.754	0.5696
Li et al	0.1294	0.0999	0.1841	0.1554	0.0667	0.1596

Los videos '*SS*', '*FT*', '*MR*', '*LB*', '*WS*' y '*CAM*' son videos que tienen fondos dinámicos, '*LB*' es un video que contiene cambios de iluminación. De acuerdo con los resultados mostrados en la Tabla 3.4, se puede observar que en general, el desempeño de SOM-CNN en fondos dinámicos tiene mejores resultados que SSOM, CNN, BNN y *Li et al*, además tiene resultados comparables con los de SOBS. Entre los métodos de ajuste automático de parámetros, SOM-CNN tiene mejores resultados en casi todos los videos.

Para entender el funcionamiento de SOM-CNN se describen a continuación los videos mostrados.

Video CAM: En este video varios árboles forman un fondo dinámico severo. Este video de 1439 cuadros tiene $\mu_{\mu d}=0.0229$ y $\mu L_d=527.39$, por lo que SOM-CNN aumenta de forma considerable los valores de α_s, σ_s, a_1 y a_2, eliminando de forma significativa el ruido causado por el fondo dinámico severo, pero generando problemas de falsos negativos. Dichos falsos negativos causan un desempeño regular haciendo que SOM-CNN tenga un desempeño por arriba de los métodos que realiza ajuste automático de parámetros y *Li et al*.

Video FT: Un escenario con una fuente y hasta dos objetos dinámicos por cuadro. Este video de 523 cuadros tiene problemas severos de fondo dinámico en la parte del escenario donde se localiza la fuente. En este video, $\mu_{\mu d}=0.163$ y $\mu L_d=316$. Con estos valores SOM-CNN incrementó los valores de α_s, σ_s, a_1 y a_2, pero no lo suficiente para eliminar el ruido de la fuente ya que esta causaba un ruido muy severo como se puede apreciar en el valor de μL_d. Además, se presentaron problemas de falsos negativos en este video por el aumento de los valores de a_1 y a_2. Por estas dos razones, SOM-CNN tuvo un desempeño regular superando solamente a *Li et al* y a BNN y empatando a SSOM. La razón por la cual SOM-CNN tuvo problemas fue porque el ruido de la fuente es muy severo, pero solo se concentra en el área de la fuente. Dicho ruido causa altos valores en μ_d aumentando significativamente el valor de los parámetros de aprendizaje. En la Figura 3.36 se observa un ejemplo del desempeño de SOM-CNN con este video.

Video LB: Un escenario con dos cambios de iluminación repentinos. Este video de 1546 cuadros tiene $\mu_{\mu d}=0.0078$ y $\mu L_d=36.53$, por lo que SOM-CNN asume que tiene condiciones normales. SOM-CNN tardó menos de diez cuadros en adaptarse a las nuevas condiciones de iluminación en ambos cambios de iluminación, mientras que el resto de los métodos necesitan hasta más de 100 cuadros en adaptarse. En [126] se puede ver un análisis de estos cambios de iluminación con este video. De acuerdo a [13], SOBS no puede modelar correctamente después de los cambios de iluminación en '*LB*'. SOM-CNN puede modelar correctamente el fondo después de cambios de iluminación, mientras que el resto de los métodos tienen problemas. Pare este video SOM-CNN queda en primer lugar como se observa en la Tabla 3.4.

Video MR: Un escenario con una cortina que se mueve eventualmente. Este video de 2964 cuadros tiene una cortina que puede causar falsos positivos en la segmentación. Dado que $\mu_{\mu d}=0.0076$ y $\mu L_d=19.27$, SOM-CNN lo tomó como un video en condiciones normales, lo cual fue un factor que mejoró los resultados de la segmentación ya que no se tuvieron problemas de falsos negativos. Solo se tenían problemas de errores causados por el movimiento de las cortinas, pero como se muestra en la Figura 3.36 estos no son tan significativos, de modo que SOM-CNN quedó en segundo lugar para este video.

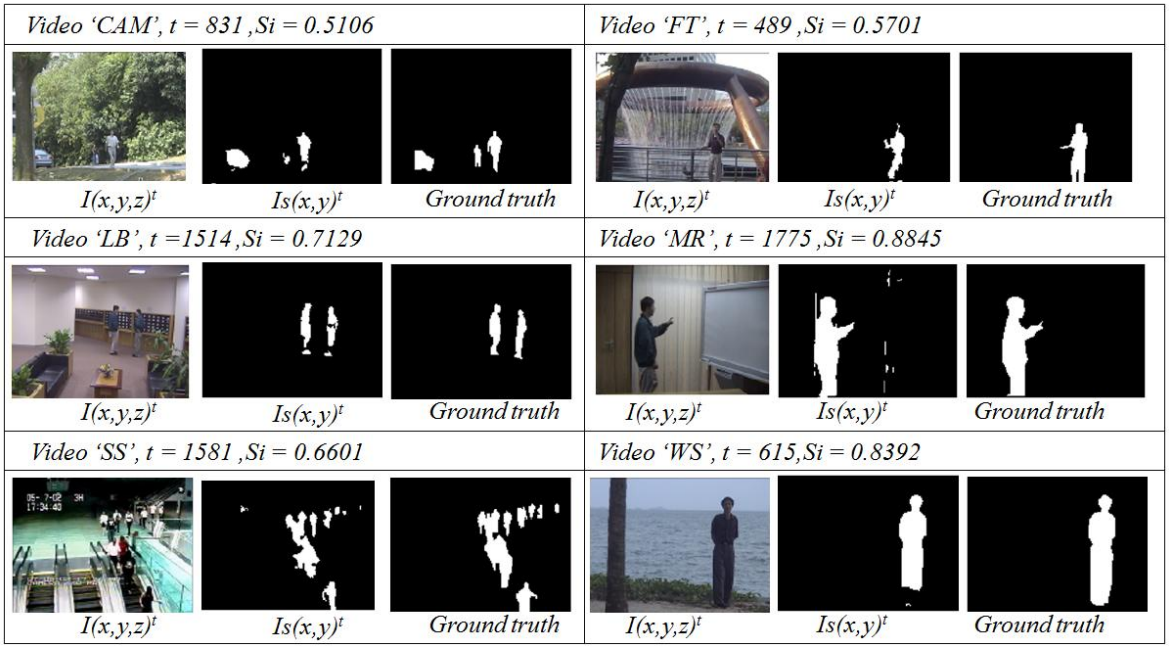


Figura 3.36. Resultados de SOM-CNN con videos utilizados en la literatura para comparar métodos.

Video SS: Un escenario con escaleras eléctricas, múltiples objetos dinámicos y eventuales cambios de iluminación. Este video de 3417 cuadros tiene problemas severos de fondo dinámico por las escaleras eléctricas, por lo tanto el promedio $\mu_{\mu d}$ de todos los valores de μ_d fue 0.0247 y el promedio μL_d de todos los valores de L_d fue 240.76. Con estos valores SOM-CNN incrementó considerablemente los valores de α_s, σ_s, a_1 y a_2, lo cual causó que se redujera el ruido por el fondo dinámico pero aumentó la cantidad de falsos negativos como se muestra en la Figura 3.36. A pesar de estos falsos negativos, SOM-CNN obtuvo el primer lugar en las comparaciones como se ve en la Tabla 3.4.

Video WS: Un escenario con mar, lo cual causa fondos dinámicos. Este video de 633 cuadros tiene $\mu_{\mu d}=0.0128$ y $\mu L_d=328.61$ es considerado por SOM-CNN como Fondo dinámico, pero el ruido no es crítico, causando que $\alpha_s \approx \alpha_o$, $\sigma_s \approx \sigma_o$, $a_1 \approx a_{10}$ y $a_2 \approx a_{20}$. Estos

valores causan ligeros problemas de falsos negativos como se ve en la Figura 3.36, pero remueven el ruido causado por el Fondo dinámico. Debido a los falsos negativos, este video queda en segundo lugar.

En la Tabla 3.5 y 3.6 se pueden observar ejemplos de mediciones de videos de otras bases de datos, y en todos los casos, SOM-CNN muestra mejores resultados que los métodos que realizan ajuste automático de parámetros. En el caso del video '*AB*' SOM-CNN tiene falsos negativos al detectar las personas como se observa en la Figura 3.35, pero SSOM y CNN tienen problemas de falsos negativos en las personas y el autobús. En el caso de '*WT*', SSOM y CNN tienen problemas considerables de falsos positivos y negativos. Para el video '*CF*' los tres métodos obtienen buenos resultados, pero SOM-CNN obtiene mejores resultados. Los resultados de '*LS*' y '*TD*' se analizarán posteriormente.

Tabla 3.5 Resultados de *P, Rc, F1* y *Si* de otros videos.

Video/Cuadro	***AB, t=590***	***CF, t=251***	***LS, t=1865***	***TD, t=1851***	***WT, t=248***
P	0.7365	0.9676	0.6297	0.8603	0.9708
Rc	0.9901	0.9324	0.9613	0.6579	0.9917
F1	0.8603	0.9496	0.761	0.7456	0.9811
Si	0.7311	0.9041	0.6145	0.5944	0.9631

Tabla 3.6 Resultados de *Si* de métodos con ajuste automático de parámetros

Método	***AB, t=590***	***CF, t=251***	***LS, t=1865***	***TD, t=1851***	***WT, t=248***
CNN	0.3912	0.8667	0.0468	0.1077	0.68
SSOM	0.2585	0.8182	0	0.2861	0.5589

Desempeño de SOM-CNN en problemas de cambios de iluminación. En las Figuras 3.37, 3.38 y 3.39 se observan los resultados de cambios de iluminación de SSOM, CNN y SOM-CNN, donde este último muestra mejores resultados. En el caso de la Figura 3.37 se retoma el ejemplo de '*TD*' donde las mejoras en los resultados de SOM-CNN se deben al manejo del análisis de escenarios obscuros de SOM-CNN, ya que los resultados bajos en SSOM y CNN se deben a camuflaje como se observa en la Figura 3.37 y en las Tablas 3.5 y 3.6. El método SOBS también reporta resultados para este método, donde *Si* fue 0.71.

Figura 3.37. Resultados de segmentación de objetos en condiciones de obscuridad y cambios de iluminación gradual. a) Video '*TD*', *t=1851*. b) $Is(x,y)^t$ con SOM-CNN. c) Resultados de segmentación con CNN. d) Resultados de segmentacón con SSOM.

Figura 3.38. Resultados de segmentación de objetos en cambios de iluminación repentinos. (a) Video '*LS*', *t=1866*. a) $Is(x,y)^t$ con SOM-CNN. b) Resultados de segmentación con CNN. c) Resultados de segmentacón con SSOM.

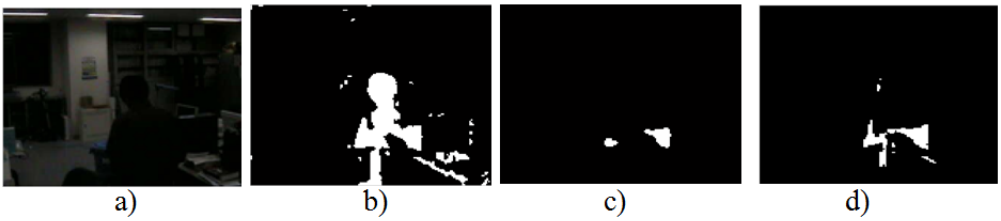


Figura 3.39. Resultados de segmentación de objetos en cambios de iluminación repentinos y escenarios obscuros. a) Video '*CM*', *t=1050*. b) $Is(x,y)^t$ con SOM-CNN. c) Resultados de segmentación con CNN. d) Resultados de segmentación con SSOM.

En la Figura 3.38 se retoma el ejemplo en '*LS*', donde los métodos SSOM y CNN no generan ningún resultado, ya que estos métodos no fueron diseñados para cambios de iluminación repentinos. El video '*LS*' no es utilizado para evaluar los métodos SSOM, CNN, SOBS, BNN y Li *et al*, ya que los cambios de iluminación repentinos (En la Figura 3.40 se observa la señal de entropía de este video), causan problemas en los resultados de segmentación en dicho métodos. Sin embargo, SOM-CNN puede manejar de manera satisfactoria estos cambios de iluminación. En la Tabla 3.7 se muestran otros métodos diseñados para cambios de iluminación [109], pero el método SOM-CNN tiene los mejores resultados. Además que los métodos mostrados en la Tabla 3.7 no soportan Fondos dinámicos.

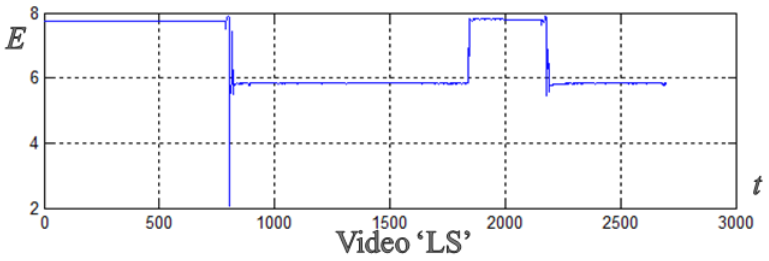


Figura 3.40. Señal de entropía del video 'LS'.

En la Figura 3.39 se observan los resultados del video '*CM*' donde se tienen cambios de iluminación repentinos y escenarios obscuros. Se puede observar en este video que SOM-CNN tiene mejores resultados que SSOM y CNN, esto debido al módulo de cambios

de iluminación de SOM-CNN. De este video no se encontró alguna referencia en la literatura. Este video tiene un alto grado de complejidad debido a que el escenario tiene cambios de iluminación repentinos y los cambios de las condiciones cromáticas son muy variantes ya que cuando las luces están encendidas, todos los objetos de colores claros generan bajo contraste con niveles de color muy claros. En obscuridad, el escenario es completamente obscuro ($\sigma<0.15$, causando contrastes muy bajos). Estas condiciones cromáticas y los cambios de iluminación hacen que este video sea muy complejo para ser procesado por los métodos que se diseñaron para fondos dinámicos.

Desempeño SOM-CNN. De todas las mediciones de desempeño realizadas, el promedio de las métricas para estos videos fue de $P = 0.875$, $Rc = 0.8316$, $F1 = 0.843$ y $Si = 0.741$. De acuerdo a las definiciones de estas métricas, SOM-CNN tiene buenos resultados eliminando más falsos positivos que falsos negativos. Es decir, SOM-CNN tiene mejor desempeño eliminando ruido que detectando objetos con problemas de camuflaje. Esta situación se debe a que algunas pruebas se realizaron utilizando videos con escenarios obscuros y en videos con fondos dinámicos severos se generan problemas de falsos negativos.

Tabla 3.7 Resultados de Si de '*LS*' con varios métodos para cambios de iluminación.

Método	**'LS'**
SOM-CNN	0.6145
ISBS	0.5993
MTD	0.1809
SSD	0.03

Como se analizó en esta subsección, SOM-CNN es un método adaptivo que tiene un desempeño comparable a los métodos de ajuste manual de parámetros o que necesitan entrenamiento previo. Sin embargo, SOM-CNN tiene mejor desempeño que los métodos que utilizan ajuste automático de parámetros.

Además, SOM-CNN tiene un desempeño similar a los métodos que se diseñaron para cambios de iluminación. Esto es una de las principales contribuciones de SOM-CNN, ya que en la literatura no se reportan métodos de segmentación de objetos dinámicos adaptivos que sean robustos a fondos dinámicos y problemas de iluminación. Para caracterizar la contribución del método propuesto, en la siguiente sección se realizará una validación utilizando la base de datos de *changedetection.net*.

3.5 Validación de SOM-CNN

En la sección anterior se presentó SOM-CNN, un método de segmentación de objetos dinámicos que surge como una aplicación de las RNA propuestas en este trabajo de tesis RESOM y NTCNN. En este estudio se pudo observar que SOM-CNN tiene un desempeño equiparable con cualquier método de segmentación de objetos dinámicos exitoso, pero adicionalmente SOM-CNN se adapta fácilmente a cambios de iluminación. Sin embargo, este estudio comparativo se realizó con videos que se utilizaron en el diseño del método de segmentación SOM-CNN. Por lo tanto, fue necesario realizar un segundo experimento con videos que no se hayan utilizado durante el diseño de los experimentos. Una base de datos que contiene suficientes videos para realizar una validación y caracterización para un método de segmentación es *changedetecion*.

3.5.1 Descripción de las bases de datos *changedetection*

Esta base contiene 31 videos con diferentes problemas que pueden ser utilizados para conocer más a fondo las debilidades y las fortalezas de SOM-CNN. Los problemas que tienen los videos incluyen fondos dinámicos, objetos dinámicos que permanecen estacionarios durante una gran cantidad de cuadros, movimiento de cámaras, sombras, reflejos, objetos dinámicos presentes en los primeros cuadros de la secuencia de video, escenarios con bajos contrastes, videos a escalas de grises, etc. Muchos de estos problemas están mínimamente presentes o no se encuentran en los videos de diseño, por lo tanto estudiar el desempeño de SOM-CNN con los videos de *changedetection* puede ser útil para entender mejor el comportamiento de dicho método. La base de datos tiene dos conjuntos: *DATASET 2012* y *DATASET 2014*. El diseño de SOM-CNN se realizó antes de iniciar el año 2014, en consecuencia se utilizó el conjunto de videos *DATASET 2012*. Este conjunto tiene seis categorías, las cuales son:

Baseline: Contiene cuatro videos que no presentan problemas significativos. Los videos son: *highway (HG)*, *office (OF)*, *pedestrian (PD)* y *PETS2006 (P6)*. El escenario de '*HG'* tiene una autopista y cruzan varios automóviles desde el primer cuadro, generando un problema de *bootstrapping*. '*OF'* es un video de una oficina en el que una persona (objeto dinámico) se mantiene estacionaria por varios cuadros. El video '*PD'* contiene un escenario muy claro de un parque donde cruzan algunas personas formándose problemas de sombras.

El escenario de '*P6*' es una sala en la que cruzan varias personas. En la Figura 3.41 se observan los cuadros iniciales de cada video.

Figura 3.41. Primer cuadro de la categoría *Baseline*.

CameraJitter: Los videos en esta categoría son: *badminton (BD), boulevard (BV), sidewalk (SD)* y *traffic (TF)*. Los cuatro videos contienen problemas fuertes causados por vibraciones en la cámara que adquiere el video. Además, los videos '*BD*', '*SD*' y '*TF*' contienen objetos dinámicos al inicio de la secuencia, por lo que adicionalmente estos videos tienen problemas de *bootstrapping*. En la Figura 3.42 se observan los cuadros iniciales de cada video.

Figura 3.42. Primer cuadro de los videos de la categoría *CameraJitter*.

DynamicBackground: Contiene seis videos con problemas de fondos dinámicos. Los videos son *boats (BT), canoe (CN), fall (FL), fountain01 (F01), fountain02 (F02)* y *overpass (OP)*. Los videos '*BT*', '*CN*' contienen como fondo dinámico un lago además de contener objetos dinámicos que se mueven lentamente. El video '*FL*' contiene un árbol que genera un fondo dinámico muy severo en una porción del escenario. Los videos '*F01*' y '*F02*' tienen fuentes y objetos dinámicos muy pequeños respecto al tamaño del escenario. El video '*OP*' tiene un árbol que genera un fondo dinámico severo. En la Figura 3.43 se observan los cuadros iniciales de cada video de esta categoría.

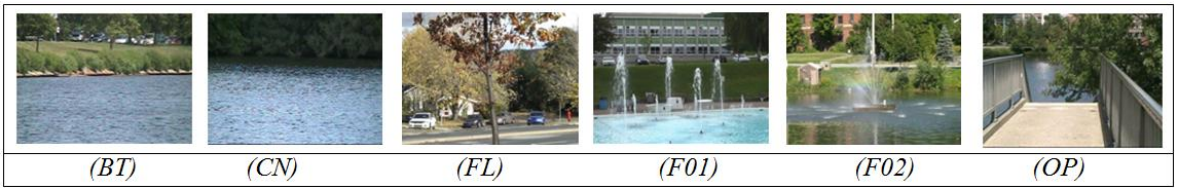


Figura 3.43. Primer cuadro de los videos de la categoría *DynamicBackground.*

IntermittentObjectMotion: Esta categoría contiene seis videos con objetos de fondo que son removidos, objetos que se anexan al fondo durante la secuencia de video, y objetos dinámicos que se mantienen estacionarios durante una gran cantidad de cuadros. Estos videos son *abandonedBox (AX)*, *parking (PK)*, *sofa (SF)*, *StreethLight (ST)*, *Trampstop (TS)* y *WinterDriveWay (WD)*. Todos los videos excepto '*SF*' son en exteriores y contienen como objetos dinámicos automóviles y personas. Los videos de '*AX*' y '*TS*' contienen además trenes. El video '*SF*' es un escenario en interior donde los objetos dinámicos son personas y cajas. En la Figura 3.44 se observan los cuadros iniciales de cada video de esta categoría.

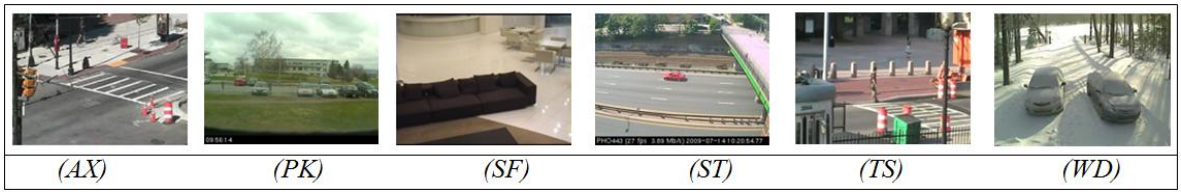


Figura 3.44. Primer cuadro de la categoría *IntermittenrObjectMotion.*

Shadow: Esta categoría contiene seis videos con problemas de sombras y reflejos. Además, algunos videos incluyen objetos dinámicos que permanecen estacionarios por una gran cantidad de cuadros. Estos videos son: *Backdoor (BC)*, *Bugalows (BW)*, *busStation (BS)*, *copyMachine (CH)*, *cubicle (CB)* y *PeopleInShade (PS)*. El video '*BC*' tiene como objetos dinámicos personas que cruzan por el escenario al igual que '*BS*', '*CH*', '*CB*' y '*PS*', además '*BC'*, '*CB*' y '*PS*' contienen cambios en los reflejos proyectados en el escenario. El caso de '*CH*' tiene personas que permanecen estacionarias por cientos de cuadros. El video '*BW*' contiene automóviles que van cruzando por el escenario. En la Figura 3.45 se observan los cuadros iniciales de cada video de esta categoría.

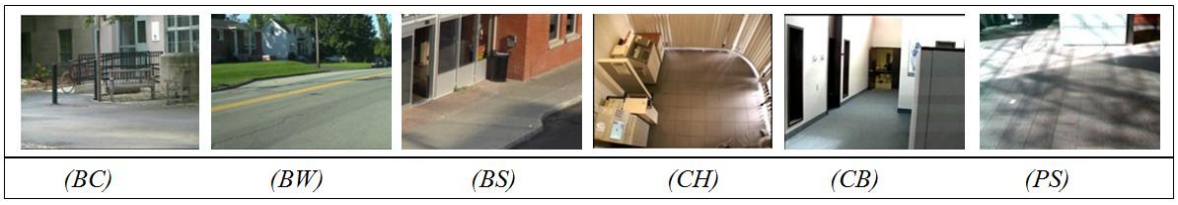


Figura 3.45. Primer cuadro de la categoría *Shadow.*

Thermal: Contiene cinco videos que fueron adquiridos por cámaras térmicas. Estos videos no contienen información de color y el contraste de estos videos es muy bajo. Los videos de esta categoría son *corridor (CR)*, *dinningRoom (DR)*, *lakeSide (LK)*, *library (LR)* y *park*

(PK). Los videos '*CR*', '*PK*' contienen como objetos dinámicos personas que cruzan por el escenario. Los videos '*DR*' y '*LR*' tienen como objetos dinámicos personas que permanecen en ocasiones en la misma posición durante varios cuadros, incluso el video '*LR*' contiene una persona que pasa cientos de cuadros en la misma posición. El video '*LK*' tiene como objetos dinámicos personas y un bote. No obstante, este último video tiene problemas severos de camuflaje. En la Figura 3.46 se observan los cuadros iniciales de cada video de esta categoría.

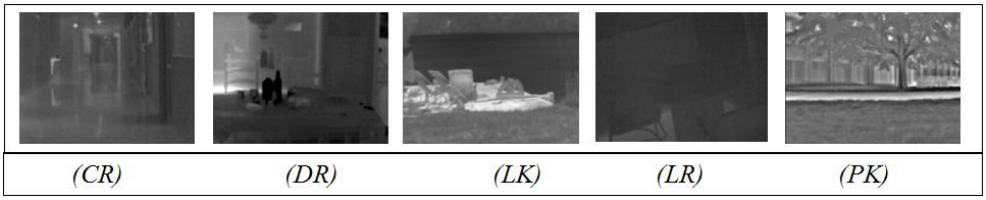


Figura 3.46. Primer cuadro de la categoría *Thermal.*

Los 31 videos de todas las categorías tienen diferentes resoluciones. Sin embargo, para este ejercicio de validación y caracterización, todos los videos se redujeron a una resolución de 120x160. Los videos fueron capturados con diferentes tipos de cámaras, diferentes condiciones de iluminación y la mayoría de los videos tienen 1000 a 7000 cuadros. Solo el video '*PK*' tiene menos de 1000 cuadros.

Un aspecto fundamental por el cual se eligió esta base de datos es porque a diferencia de las otras bases de datos, se tiene disponible un *ground truth* para cada cuadro. Para estos videos, los *Ground truth* tienen cinco clases: pixel de fondo (negro), pixel de objeto dinámico (blanco), pixel de no interés (gris medio) o fuera de la ROI (*Region Of Interest*), pixel de clase desconocida (gris-claro) y pixel de sombra (región de gris medio bordeada por pixeles de clase desconocida). En la Figura 3.47 se pueden observar todos los estados de los *ground truth* en un cuadro del video '*BW*'. En los videos existen al menos las clases Fondo, Objeto dinámico y fuera de la ROI. Todos se dividen en su fase de entrenamiento y su fase de evaluación. En la fase de entrenamiento el desarrollador puede utilizar estos cuadros para entrenar su método, ya que todos los pixeles están como pixel de no interés. La siguiente fase es la de evaluación y se utiliza para medir el desempeño del método propuesto.

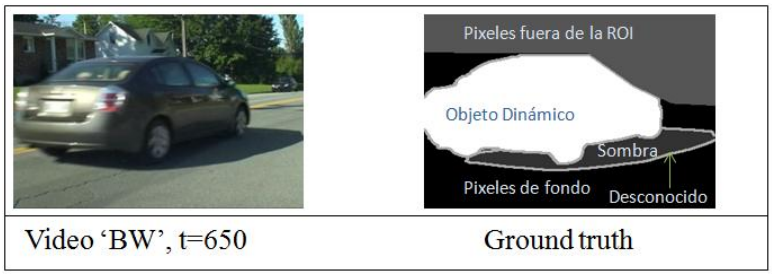


Figura 3.47. Clases de un *Ground truth* para los videos de *changedetection.*

La idea principal de esta base de videos es probar cada método de segmentación en un concurso para obtener un lugar entre cada método. Sin embargo, en este trabajo de tesis esta base de videos se utilizó con motivos únicamente de validación y caracterización del método propuesto y no para participar activamente en dicha contienda. Es decir, se obtuvieron mediciones del desempeño de SOM-CNN con todos los videos de *changedetection*, se analizaron los resultados y se compararon solo con los métodos más exitosos en esta base de datos. Las métricas para esta base de datos son *P*, *Rc*, *F1* y adicionalmente las siguientes:

FPR: Tasa de falsos positivos, se refiere a la cantidad total de falsos positivos resultantes en un cuadro entre la cantidad de pixeles que deben ser parte del fondo. Esta dada por:

$$FPR = \frac{FP}{FP + TN} \tag{3.64}$$

FNR: Tasa de falsos negativos, se refiere a la cantidad total de falsos negativos resultantes en un cuadro entre la cantidad de pixeles que deben ser parte de los objetos dinámicos. Esta dada por:

$$FNR = \frac{FN}{FN + TP} \tag{3.65}$$

PWC: Porcentaje de clasificaciones erróneas. Se refiere a la cantidad de pixeles clasificados erróneamente entre los pixeles de todas las clases. Esta métrica se expresa como un porcentaje dado por:

$$PWC = \frac{100(FP + FN)}{FP + FN + TP + TN} \tag{3.66}$$

Specifity (*Sp*): Se refiere a la cantidad de pixeles de fondo clasificados correctamente. Esta dado por:

$$Sp = \frac{TN}{TN + FP} \tag{3.67}$$

Para *changedetection*, las métricas para la medición del desempeño son *P*, *Rc*, *F1*, *Sp*, *FPR*, *FNR* y *PWC*. Si se analizan las métricas de *FPR*, *FNR* y *PWC*, son cero si el resultado de la segmentación coincide con el *ground truth*, mientras que *P*, *Rc*, *F1* y *Sp* deben ser uno si el resultado de la segmentación coincide con el *ground truth*. A continuación se analizarán los resultados de la experimentación con esta base de datos y la comparación del funcionamiento de SOM-CNN y los métodos más exitosos.

3.5.2 Resultados de la experimentación de SOM-CNN con los videos de *changedetection*

Para utilizar los videos de *changedetection*, se realizaron algunos cambios en el tercer módulo de SOM-CNN. Para utilizar esta base de datos existe la regla que el método a evaluar debe ajustarse automáticamente. No obstante, al analizar la parte de los videos de entrenamiento, fue posible notar que entre un cuadro y otro existe mucho más parecido entre sí que en los videos de las otras bases de datos. Esto quizás se deba a que los videos adquiridos para las categorías de *Baseline*, *cameraJitter*, *dynamicBackground*, *intemittentObjectMotion*, *Shadow* y *Thermal* fueron adquiridos con *frame rates* mayores a 25 fps. Esto se puede notar en que los videos de las bases descritas en la sección 3.1 para diseño, tienen objetos dinámicos que cruzan por los escenarios durante algunas decenas de cuadros, mientras que en *changedetection* duran a veces cientos de cuadros. Este posible *frame rate* más alto que en los videos de diseño puede causar errores en la sensibilidad de SOM-CNN para anexar al fondo objetos que permanecen estacionarios por cientos de cuadros o en objetos que se mueven lentamente. Este aspecto puede afectar más en los videos con fondos dinámicos ya que el aprendizaje de la RESOM puede resultar muy rápido si α y σ_β mantienen sus valores mayores a los establecidos en la sección 3.3. Por lo tanto, se realizó un análisis en la parte de entrenamiento de los videos con fondos dinámicos (*DynamicBackground* y *CameraJitter*) para determinar los parámetros asociados al aprendizaje α_o, σ_o y T_f. Con base a dicho análisis, se determinó que $\alpha_o = 0.1$ y $\sigma_o = 0.2$ y T_f se cambió a 15 para condiciones normales y 20 para condiciones de *bootstrapping*, aunque esto último no tiene efecto en la medición de los resultados ya que en los primeros cuadros de cada video no se mide el desempeño.

Con estas modificaciones se realizó una experimentación de SOM-CNN con todos los videos de la base de datos de *changedetection*, y en la Tabla 3.8 se pueden observar los resultados. De acuerdo a las definiciones, las métricas que están relacionadas con problemas de falsos negativos son *Rc* y *FNR*. Las métricas que están relacionadas con los falsos positivos son *P* y *FPR*. Las métricas de *PWC* y *F1* están relacionadas a la correcta detección de los objeto dinámicos, ya que incluye medición de falsos positivos y negativos. La métrica de *Sp* mide la correcta detección del fondo, por lo tanto está muy relacionado a *FPR*.

Tal como se esperaba, la categoría en donde se obtuvieron los mejores resultados es *Baseline*, donde los videos no contienen problemas significativos. La categoría donde se obtuvieron los resultados más bajos fue la de *CameraJitter*.

Como se puede observar en la Tabla 3.8, los mejores resultados en *Rc* y *FNR* fueron todos los videos de *Baseline*, '*AX*', '*ST*', '*BW*', '*CH*' y '*LR*' siendo el resultado más alto '*ST*'. Para estas métricas los resultados más bajos fueron para los videos '*F01*', '*SD*', '*LK*', '*TS*' y '*PR*'. Con base a estos resultados, se puede observar que los videos de la categoría *Baseline* concentran buenos resultados para *Rc* y *FNR*, las demás categorías concentran resultados altos y bajos. Por ejemplo, la categoría *intermittentObjectMotion* tiene el resultado más alto en '*ST*' y algunos de los más bajos. En la Figura 3.48a se puede observar un resultado con buenos niveles de *Rc* y *FNR*, donde el objeto dinámico (persona) se mantiene segmentado aún cuando se mantiene en la misma posición después de aproximadamente 2400 cuadros. En la Figura 3.48b se observa un buen resultado de Rc y FNR en un video con problemas de sombras y objetos dinámicos que permanecen estacionarios.

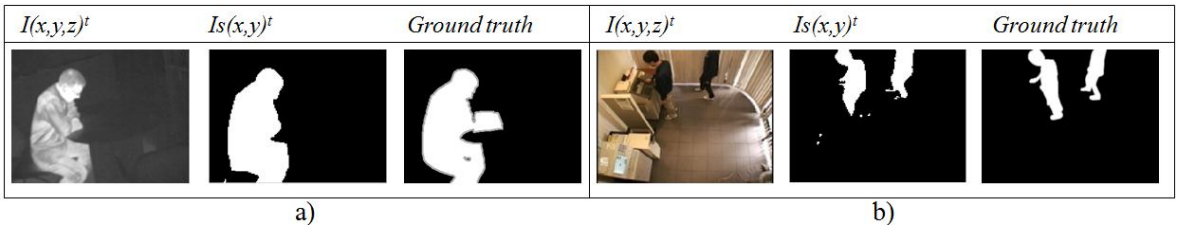


Figura 3.48. Resultados de segmentación con buen *Rc* y *FNR*. a) Video '*LR*', *t=4000*. b) Video '*CH*', *t=1600*.

Tabla 3.8 Resultados del SOM-CNN en los videos de *changedetection*.

Video	Rc	Sp	FPR	FNR	PWC	P	F1
Baeseline							
HG	0.94645	0.9926	0.00739	0.0535	0.9756	0.8739	0.9088
OF	0.97616	0.9981	0.00188	0.0238	0.3197	0.9707	0.9734
PD	0.98039	0.9996	0.00045	0.0196	0.0565	0.9316	0.9554
P6	0.909	0.9979	0.00215	0.091	0.2971	0.7986	0.8502
CameraJitter							
BD	0.46993	0.999	0.00098	0.5301	1.4324	0.9252	0.6233
BV	0.44688	0.997	0.00301	0.5531	2.6845	0.8704	0.5906
SD	0.33686	0.9969	0.00312	0.6631	1.5852	0.6795	0.4504
TF	0.46564	0.9904	0.0096	0.5344	4.0526	0.7524	0.5753
DynamicBackground							
BT	0.63761	1	0	0.3624	0.1762	0.9938	0.7768
CN	0.85545	1	0	0.1445	0.4238	0.9995	0.9219
FL	0.57533	0.9881	0.01191	0.4247	1.8795	0.4503	0.5052
F01	0.08564	0.9997	0.00035	0.9144	0.0853	0.1193	0.0997
F02	0.80659	0.9999	0	0.1934	0.0355	0.9619	0.8774
OP	0.7738	0.9984	0.00158	0.2262	0.4124	0.8485	0.8094
IntObjectMotion							
AX	0.97015	0.911	0.08901	0.0299	8.7255	0.2503	0.3979
PR	0.39491	0.937	0.06298	0.6051	9.1596	0.259	0.3128
SF	0.73177	0.9968	0.0032	0.2682	1.2493	0.8926	0.8042
ST	0.99216	0.9938	0.00618	0.0078	0.6229	0.8415	0.9106
TS	0.39115	0.9653	0.03468	0.6089	11.132	0.6347	0.484
WD	0.70379	0.932	0.06798	0.2962	6.932	0.0575	0.1064
Shadow							
BC	0.63294	0.9819	0	0.3671	2.3747	0.366	0.4638
BW	0.91558	0.9809	0.01911	0.0844	2.2833	0.7433	0.8205
BS	0.87051	0.9895	0.01047	0.1295	1.4018	0.7186	0.7873
CH	0.91143	0.9887	0.01129	0.0886	1.6077	0.8421	0.8754
CB	0.53791	0.9844	0.01559	0.4621	2.3083	0.3708	0.439
PS	0.58423	0.9951	0.00493	0.4158	2.6473	0.8678	0.6983
Thermal							
CR	0.89434	0.996	0.00403	0.1057	0.6962	0.8684	0.8812
DR	0.80226	0.9888	0.01115	0.1977	2.5963	0.8612	0.8307
LK	0.38301	0.9927	0.00734	0.617	1.569	0.4204	0.4009
LR	0.94631	0.9925	0.00745	0.0537	1.5921	0.9661	0.9561
PK	0.55881	0.9989	0.00105	0.4412	0.8099	0.896	0.6883

De acuerdo a la Tabla 3.8, la métrica *Sp* tiene buenos resultados en la categoría de *dynamicBackground* y tiene los resultados más bajos en la categoría de *intermittentObjectMotion*. El resto de las categorías generan resultados altos y bajos, pero en general, existe muy poca variación en esta métrica. Para las métricas de *P* y *FPR* los mejores resultados se concentran en *DynamicBackground* aunque también se presentan algunos videos de *Baseline* y *Shadow*. Un caso interesante es el de '*BC*' el cual es un video que contiene varios reflejos que van cambiado de manera gradual. Estos reflejos cambiantes deberían causar falsos positivos, sin embargo, tiene el primer lugar en *FPR*. Esto se debe a que el módulo de sombras y reflejos funciona adecuadamente para este video. Una situación similar sucede con el video '*PS*'. No obstante, el video '*CB*' también

tiene muchos reflejos de sol que van cambiando las condiciones del escenario de manera severa, por lo que el ajuste en reflejos y sombras no funciona bien en este video causando bajos resultaos en *P* y *FPR*. En la Figura 3.49a se observa un resultado del video '*CN*', donde la canoa fue segmentada exitosamente, pero las personas se perdieron porque a_1 y a_2 tienen valores muy altos ya que el fondo dinámico causa niveles altos en μ_d, el cual a su vez causa que $a_1=0.1557$ y $a_2=0.1073$ (ver ecuaciones 3.44 y 3.45). En la Figura 3.49b se puede observar el resultado de la segmentación de un cuadro de la secuencia '*BC*' donde existen varias transiciones en la luz reflejada en el piso. En este último caso, SOM-CNN maneja adecuadamente los parámetros para que las transiciones de luz no causen ruidos en la segmentación como se ve en la Figura 3.49b en $Is(x,y)^t$.

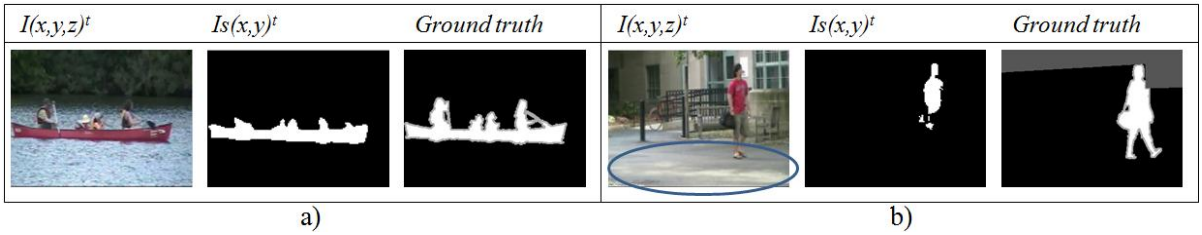


Figura 3.49. Resultados de segmentación con buen Rc y FNR. a) Video '*CN*', t=955. b) Video '*BC*', t=1745. El área encerrada en círculo tiene varias transiciones en luminancia por los reflejos cambiantes del sol.

Las métricas *PWC* y *F1* tienen información relacionada con los falsos positivos y negativos, por lo tanto estas métricas pueden describir de mejor manera el desempeño de SOM-CNN que las anteriores. Por lo tanto, para estas métricas se muestran las gráficas del mejor al peor resultado en las Figuras 3.50 y 3.51. Se puede observar en la Figura 3.50 y 3.51 que tanto los peores como los mejores resultados no tienen una categoría definida y también se puede observar que en ambas métricas no existe una relación entre si ya que algunos de los videos con los peores resultados en *F1* tienen buenos resultados en *PWC*, como es el caso del video '*F01'*. También algunos de los videos con buenos resultados en *F1* tienen buenos resultados en *PWC* como el video '*PD*'. Esto se debe a la manera en como el tercer módulo sintoniza los parámetros de DR-SOM y NTCNN, lo cual se explica a continuación por categoría.

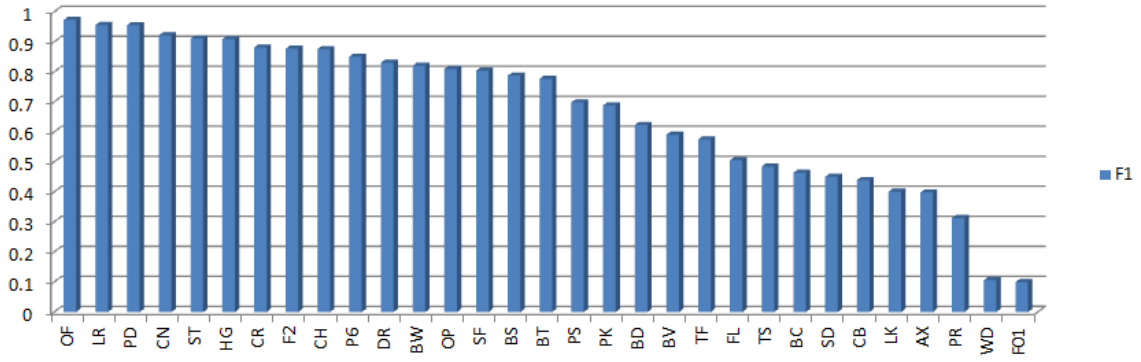


Figura 3.50. Resultados de F1 del mejor al peor.

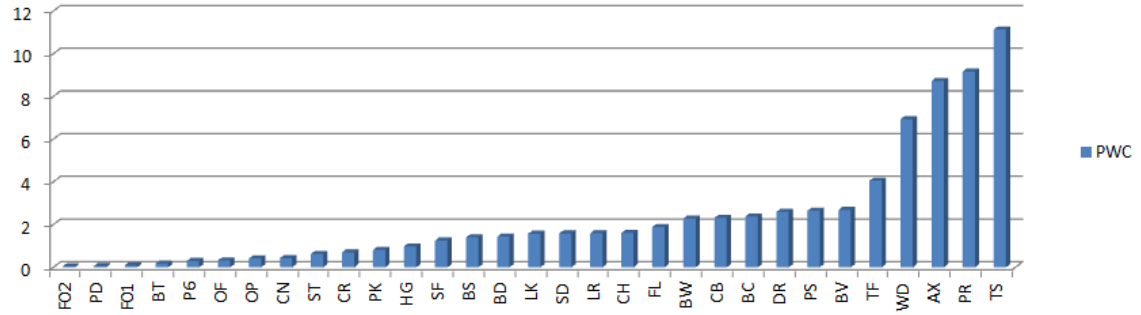


Figura 3.51. Resultados de PWC del mejor al peor.

Baseline. Los videos '*HG*', '*OF*', '*PD*' y '*P6*' en esta categoría son diagnosticados por el tercer módulo como "condiciones normales", por lo que α, σ_β, se mantienen cercanos a cero y a_1, a_2 no varían mucho. Como existen muy pocos cambios en el fondo de estos videos, los resultados fueron satisfactorios en esta categoría tanto en *F1* y *PWC* como se observa en la Figura 3.50 y 3.51. En la Figura 3.52a se observa el ejemplo del video '*OF*' donde el resultado fue bastante satisfactorio de acuerdo a *F1* y *PWC* a pesar de que la persona cambia muy poco de posición en los anteriores 900 cuadros.

cameraJitter: Todos los videos de esta categoría ('*BD*', '*BV*', '*SD*' y '*TF*') son diagnosticados por el tercer módulo como "fondos dinámicos". El movimiento de la cámara causa valores de α, σ_β, a_1 y a_2 cercanos a las magnitudes máximas definidas por las ecuaciones 3.42 a 3.44 ($\alpha \approx 0.35$, $\sigma_\beta \approx 0.9$, $a_1 \approx 0.2$ y $a_2 \approx 0.15$). Esto causa para los videos de '*BD*', '*BV*', '*SD*' y '*TF*' buenos resultados reduciendo ruido, pero son los videos con más problemas de falsos negativos. Por lo tanto, los resultados en esta categoría son regulares como se ve en las Figuras 3.50 y 3.52, y en la Figura 3.52b se observa que aunque no se presenta ruido por el movimiento de la cámara, existen problemas de falsos negativos.

dynamicBakcground: Todos los videos de esta categoría (‘*BT*’, ‘*CN*’, ‘*FL*’, ‘*F01*’, ‘*F02*’ y ‘*OP*’) son diagnosticados por el tercer módulo como “fondos dinámicos”. Para ‘*CN*’ los valores de α, σ_β, a_1 y a_2 se van a las magnitudes máximas definidas por las ecuaciones 3.42 a 3.44. El resto de los videos tienen variaciones entre $\alpha \approx 0.1$, $\sigma_\beta \approx 0.2$, $a_1 \approx 0.08$, $a_2 \approx 0.06$ a $\alpha \approx 0.35$, $\sigma_\beta \approx 0.9$, $a_1 \approx 0.2$, $a_2 \approx 0.15$. Esto causa el video ‘*CN*’ resultados satisfactorios, para ‘*OP*’, ‘*BT*’ y ‘*F02*’ los resultados son regulares. En los casos de estos últimos, los resultados regulares son causados por los falsos negativos en los objetos dinámicos, como se ve en el ejemplo de la Figura 3.52c. ‘*FL*’ también tiene resultados regulares, pero eso se debe a que el ruido aunque es muy severo, cubre solo ¼, por lo que μ_d no varía mucho, causando problemas de ruido, lo cual se aprecia en las métricas de *P* y *FPR*. El video ‘*F01*’ tiene el peor resultado en *F1*, pero uno de los mejores en *PWC*; esto se debe a que SOM-CNN elimina correctamente el ruido haciendo que la mayor cantidad de los pixeles estén bien diagnosticados causando un alto valor en *PWC*. Pero debido al tamaño tan reducido del objeto dinámico y a su similitud con el color del fondo, este no se puede segmentar causando que $Rc \approx 0$ y por lo tanto $F1 \approx 0$.

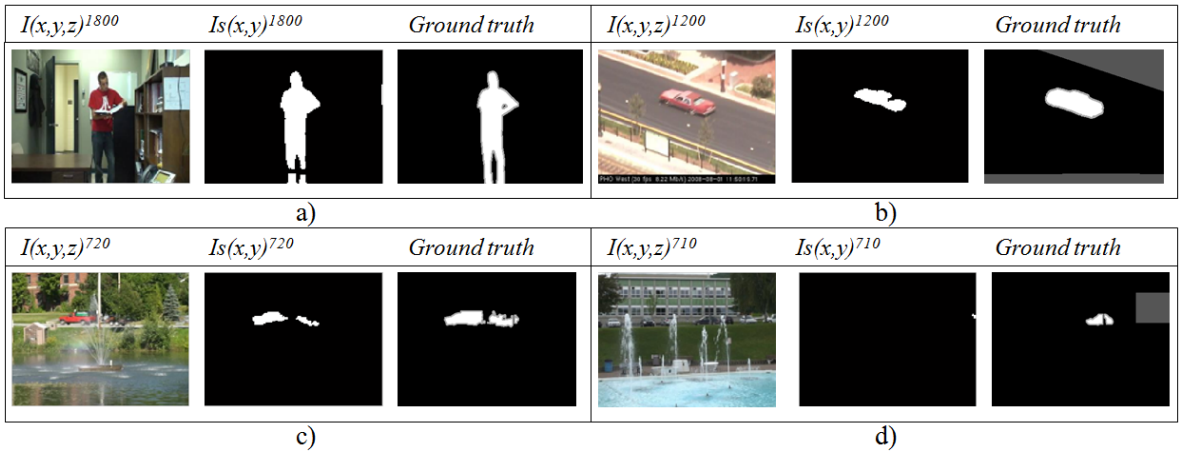


Figura 3.52. Resultados de segmentación. a) Video ‘*OF*’. b) Video ‘*BV*’. c) Video ‘*F02*’. d) Video ‘*F01*’.

intermittentObjectMotion: SOM-CNN arroja resultados interesantes con esta categoría, ya que son muy variados. Todos los de esta categoría (‘*AX*’, ‘*PR*’, ‘*SF*’, ‘*ST*’, ‘*TS*’ y ‘*WD*’) fueron interpretados por el tercer módulo como condiciones normales. SOM-CNN genera resultados satisfactorios con ‘*ST*’, pero en ‘*SF*’ genera resultados regulares, lo cual se debe a problemas de camuflaje como se ve en la Figura 3.53a. El video ‘*PK*’ tiene resultados regulares por los mismos problemas de camuflaje. Los videos ‘*AX*’, ‘*TS*’ y ‘*WD*’ no generan buenos resultados, lo cual se debe a que muchos objetos que originalmente fueron

de fondo son retirados o se mueven de posición. En condiciones normales la actualización de fondo tarda varios cuadros causando errores de segmentación. Una forma de resolver esto sin generar errores de segmentación cuando objetos dinámicos que permanecen estacionarios es sintonizar el parámetro s_2 cuando SOM-CNN está en condiciones normales. Un aspecto que se debe considerar es que los videos '*AX*', '*TS*' tienen la mayor parte de la escena fuera de la ROI como se ve en el ejemplo de '*AX*' de la Figura 3.53b, por lo cual este análisis no considera la segmentación de algunos elementos como el tren y algunas personas y objetos de estos escenarios. En todos los videos algunos objetos dinámicos permanecen por varios cuadros, pero esto no genera errores de segmentación.

shadow: Todos los de esta categoría ('*BC*', '*BW*', '*BS*', '*CH*', '*CB*' y '*PS*') son diagnosticados por el tercer módulo como "condiciones normales". En los videos '*BC*', '*CB*' y '*PS*' algunos cuadros cambian su estado a condiciones normales con sombras y reflejos. De acuerdo a las métricas, '*BS*' y '*CH*' tiene resultados satisfactorios, mientras que '*BC*', '*BW*' y '*PS*' tienen resultados regulares. Los videos '*BC*', '*PS*' tienen problemas de camuflaje cuando SOM-CNN detecta cambios en los reflejos y sombras. *BW* tiene bajos resultados en *P* y *PWC* a causa de sombras como se puede observar en la Figura 3.53c. El video '*CB*' generó resultados bajos debido como se mencionó anteriormente a cambios severos de reflejo de luz solar.

Thermal: Todos los de esta categoría ('*CR*', '*DR*', '*LK*', '*LR*', y '*PK*') son diagnosticados por el tercer módulo como "escenarios obscuros y de bajos contrastes" ya que todos los videos están en escalas de grises. Esto permitió reducir de manera significativa los problemas de camuflaje, generando buenos resultados en los videos '*CR*', '*LR*'. En '*DR*' los resultados son un poco más regulares, lo cual es causado porque objetos que originalmente son de fondo cambiaron de posición y SOM-CNN tardó decenas de cuadros para actualizar el fondo. No obstante, los videos '*PK*' y '*LK*' fueron afectados por camuflaje, incluso el video '*LK*' al tener un camuflaje severo, la segmentación no fue satisfactoria como se observa en la Figura 3.53d donde el objeto dinámico esta dentro del círculo.

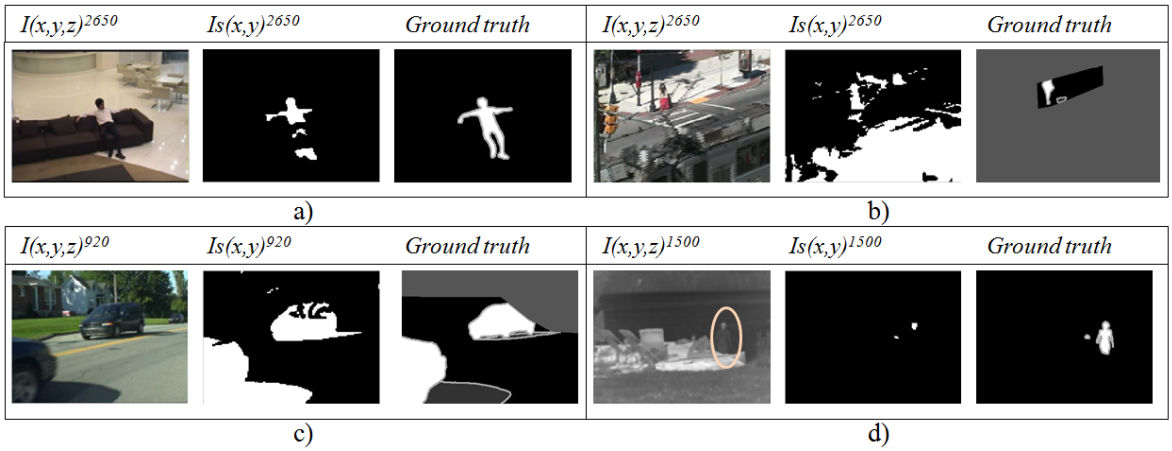


Figura 3.53. Resultados de segmentación. a) Video '*SF*'. b) Video '*AX*'. c) Video '*BW*'. d) Video '*LK*'.

3.5.3 Análisis de la experimentación de SOM-CNN con los videos de *changedetection*

De acuerdo a la evaluación del tercer módulo de SOM-CNN, los videos de las categorías *Baseline*, *intermittentObjectMotion* y *shadow* están en condiciones normales, los videos de las categorías *cameraJitter* y *dynamicBackground* tienen fondos dinámicos, los videos de *thermal* tienen condiciones de bajo contraste y los videos '*BC*', '*CB*' y '*PS*' de *shadow* tienen reflejos. De acuerdo a la definición de cada categoría en *changedetetion.net*, estos diagnósticos realizados por el tercer módulo de SOM-CNN son los más correctos tomando como base que este módulo clasifica el estado de un escenario en "condiciones normales", "condiciones normales con sombras y reflejos", "fondos dinámicos", "escenarios de bajo contraste", "cambios de iluminación gradual o repentino" y "*bootstrapping*".

En la Tabla 3.9 se puede observar la media μ_{ch} y la desviación estándar σ_{ch} de cada métrica de la evaluación de SOM-CNN con cada categoría y todos los videos de *changedetection*. Respecto a las métricas, se puede observar que *Rc*, *Sp*, *FNR*, *PWC*, *P* y *F1* tuvieron el mejor resultado en *Baseline* y *FPR* tuvo el mejor resultado en *dynamicBackground*. El peor resultado para *Rc* y *FNR* cayó en la categoría de *cameraJitter*, para *Sp*, *FPR*, *PWC*, *P* y *F1* fue en *intermittentObjectMotion*. Con estos resultados, es posible concluir que SOM-CNN tuvo problemas de falsos en videos de la categoría *intermittentObjectMotion*, mientras que en *cameraJitter* SOM-CNN tuvo problemas de falsos negativos.

Tabla 3.9. Media μ_{ch} y desviación estándar σ_{ch} de cada video de la evaluación de SOM-CNN con los videos de *changedetection*. El resultado de las métricas de todos los videos está en la Tabla 3.8.

		Rc	**Sp**	**FPR**	**FNR**	**PWC**	**P**	**F1**
Baseline	μ_{ch}	0.953	0.99703	0.00297	0.047	0.41224	0.89371	0.92194
	σ_{ch}	0.03299	0.00304	0.00304	0.03299	0.39402	0.07484	0.05504
cameraJitter	μ_{ch}	0.42983	0.99582	0.00418	0.57017	2.43867	0.8069	0.55989
	σ_{ch}	0.06278	0.00374	0.00374	0.06278	1.21194	0.11139	0.07567
dynamicBackground	μ_{ch}	0.6224	0.99768	0.00231	0.3776	0.50211	0.72889	0.66508
	σ_{ch}	0.28345	0.00474	0.00474	0.28345	0.69412	0.36364	0.31291
IntObjectMotion	μ_{ch}	0.69732	0.956	0.044	0.30268	6.30363	0.48927	0.50267
	σ_{ch}	0.26374	0.03505	0.03505	0.26374	4.37151	0.3477	0.30391
Shadow	μ_{ch}	0.7421	0.98676	0.01023	0.2579	2.10383	0.65145	0.68073
	σ_{ch}	0.17538	0.00537	0.00695	0.17538	0.48616	0.22642	0.18687
Thermal	μ_{ch}	0.71695	0.9938	0.0062	0.28305	1.45268	0.80243	0.75143
	σ_{ch}	0.23868	0.00383	0.00383	0.23868	0.76259	0.21752	0.21902
Todos los videos	$\boldsymbol{\mu_{ch}}$	**0.6931**	**0.9865**	**0.0129**	**0.3069**	**2.3266**	**0.7107**	**0.6702**
	$\boldsymbol{\sigma_{ch}}$	**0.2409**	**0.0215**	**0.0217**	**0.2409**	**2.8287**	**0.2766**	**0.2482**

Con base a μ_{ch} de todos los videos, se puede caracterizar el desempeño; las métricas con menor error son *Sp* (ecuación 3.67), *PWC* (ecuación 3.66) y *FPR* (ecuación 3.64), mientras que las métricas con mayor error son *Rc* (ecuación 3.61), *P* (ecuación 3.60), *F1* (ecuación 3.62) y *FNR* (ecuación 3.65). De acuerdo a las definiciones, *Sp* está asociado al correcto modelado del fondo, *PWC* la tasa de pixeles erróneamente clasificados y *FPR* mide la tasa de pixeles erróneamente clasificados como parte de objetos dinámicos. Las definiciones de *P*, *Rc* y *F1* se relacionan más con la correcta detección del objeto dinámico y *FNR* mide la taza de pixeles erróneamente clasificados como parte del fondo. Con base a estas definiciones, se puede concluir que SOM-CNN tiene poco error en el modelando el fondo, pero tiene errores en la detección de objetos dinámicos. Esto no quiere decir que SOM-CNN tenga problemas de segmentación, ya que si el modelado de fondo es correcto, entonces se puede obtener una relación entre pixeles de fondo o de objeto dinámico que permita una segmentación coherente. No obstante, los errores en detección de objetos dinámicos indica que SOM-CNN obtiene resultados que no siempre representa en toda su estructura geométrica un objeto dinámico; por ejemplo, si es segmentada una persona, los resultados pueden representar coherentemente el cuerpo pero perder información de las piernas o brazos, si es segmentado un automóvil, es posible que las llantas no sean segmentadas.

De acuerdo al análisis estadístico de todos los videos, mostrado en la Tabla 3.9, las métricas se encuentran en los intervalos aproximados definidos por los límites mínimo y máximo que se muestran en la Tabla 3.10. En estos intervalos definidos por los valores estadísticos de μ_{ch} y σ_{ch} se puede notar que SOM-CNN tiene mejores resultados en reducir

falsos positivos que reduciendo falsos negativos. Esto es evidente al analizar los intervalos definidos por los límites de *Rc*, *FNR*, *FPR* y *P*; el intervalo de *Rc* tiene un rango de 0.4818 con un máximo de 0.9303 y el rango de *FNR* es de 0.4818 con un mínimo de 0.06597, mientras que *P* tiene un rango de 0.55324 con un máximo de 0.98734 y *FPR* tiene un rango de 0.04331 con un mínimo de 0. Con estos rangos y valores, se puede observar que P y FPR se acercan más a sus valores ideales[19] (P=1, FPR=0) que *Rc* y *FNR*. Esto coincide con lo analizado en la sección 3.4, ya que con base a los valores promedio de las métricas obtenidas con los videos de diseño (*P = 0.875*, *Rc = 0.8316*, *F1 = 0.843* y *Si = 0.741*) se concluyó lo mismo.

Los valores de la Tabla 3.10 indican que las métricas *Sp*, *FPR* y *PWC* pueden llegar a tener resultados iguales a sus valores ideales, de manera que de acuerdo a estos valores estadísticos, SOM-CNN puede llegar a modelar correctamente el fondo. Los valores de *Rc*, *P*, *F1* y *FNR* no llegan a sus valores ideales, pero pueden acercar bastante a estos, de manera que SOM-CNN puede llegar a representar de manera satisfactoria el objeto dinámico.

Tabla 3.10 Valores máximo y mínimos definidos por μ_{ch} y σ_{ch} de todos los videos.

	Rc	**Sp**	**FPR**	**FNR**	**PWC**	**P**	**F1**
Mínimo	0.45222	0.96501	-0.0088	0.06597	-0.50206	0.4341	0.42194
Máximo	0.93403	1.00809	0.03452	0.54778	5.15525	0.98734	0.9184

De acuerdo a la Tabla 3.9, los videos que causan que SOM-CNN tenga resultados en el límite más alejado del valor ideal de cada métrica son los videos contenidos en *cameraJitter*, *intermittentObjectMotion* y *dynamicBackground*. Algunos problemas presentes en ciertos videos causan el sesgo no deseado. En el caso de *intermittentObjectMotion* los videos que causan este sesgo son aquellos donde SOM-CNN no actualizó el fondo de manera rápida cuando objetos que eran de fondo fueron removidos o cambiaron de posición generando falsos positivos que afectaron en *P*, *F1* y *FPR*. Además hubo problemas de falsos negativos causados por algunos problemas de camuflaje que afectaron en *Rc* y *F1*. En el caso de *cameraJitter* y *dynamicBackground* el principal problema se centró en la cantidad de falsos negativos causados por los valores de a_1 y a_2 que fueron sintonizados al detectarse fondos dinámicos. Para *dynamicBackground* también

[19] Valor ideal se refiere a los valores donde el *Ground truth* y el resultado es el mismo

afectó el video de '*F01*' donde los objetos dinámicos no fueron segmentados a causa del camuflaje y lo alejados que estaban de la cámara[20]. Las categorías de *shadow* y *thermal* en general tuvieron buenos resultados, pero fueron afectadas principalmente por los resultados de los videos '*BC*', '*CB*' y '*LK*'.

3.5.4 Validación y caracterización de SOM-CNN

Con base a los resultados y el análisis de la evaluación de SOM-CNN con los videos de *changedetection* en esta subsección se resumirán las conclusiones del desempeño de este método de segmentación de objetos dinámicos.

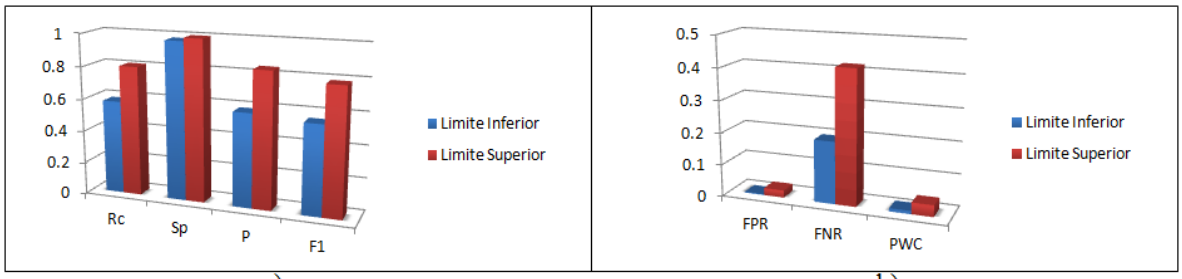
Validación con media poblacional de las mediciones del desempeño de SOM-CNN. Por su tamaño, la base de datos de *changedetection* brinda la oportunidad de realizar un análisis estadístico muy completo del funcionamiento de SOM-CNN ya que los 31 videos contienen miles de cuadros con los que se puede medir el desempeño de este método.

Para realizar una validación del desempeño obtenido en este análisis estadístico, se realizó una estimación poblacional de los valores esperados μ_{ch} obtenidos por las métricas. Para ello, se obtuvo el intervalo de confianza de cada métrica para poder conocer el rango en el cual se espera que SOM-CNN arroje resultados en evaluaciones futuras y tener una idea del comportamiento de dicho método. El intervalo de confianza estima el valor poblacional de una media a partir de los parámetros estadísticos de una muestra [127]. Esta medición está dada por:

$$\left[\mu_{ch} - z_{\alpha/2} \frac{\sigma_{ch}}{\sqrt{31}} > \mu_z > \mu_{ch} + z_{\alpha/2} \frac{\sigma_{ch}}{\sqrt{31}} \right] \tag{3.67}$$

donde $z_{\alpha/2}$ es el porcentaje de confianza, el 31 se es el número de muestras del experimento y μ_x se refiere al valor esperado poblacional. Los resultados obtenidos a 99% de confianza fueron: *Rc* [0.58134, 0.8095], Sp [0.9765, 0.9965], P [0.5823, 0.8391], F1 [0.5549, 0.7853], FPR [0.00282, 0.0229], FNR [0.19508, 0.4185] y PWC [1.014, 3.64]. En la Figura 3.54 se observan las proporciones de los intervalos de confianza para las métricas cuyo valor ideal es uno (Figura 3.54a) y para los videos cuyo valor ideal es cero (Figura 3.54b).

[20]En algunas pruebas con videos de diseño, SOM-CNN puede segmentar objetos de un tamaño muy reducido respecto al escenario, pero si se anexa el problema de camuflaje, entonces es difícil lograr la segmentación.



a) b)

Figura 3.54. Intervalos de confianza. a) Métricas cuyo valor ideal es uno. b) Métricas cuyo valor ideal es cero.

Los valores esperados poblacionales de *Rc*, *P* y *F1* son menores a los obtenidos en la sección 3.4 ya que en esa sección se utilizaron los videos para diseño. Los intervalos de confianza reafirman lo que ya se había mencionado antes; *Sp*, *FPR* y *PWC* tienen menor incertidumbre en la estimación del valor esperado poblacional y se acercan mucho a los valores ideales. Mientras que *Rc, P* y *F1* tienen un mayor nivel de incertidumbre en la estimación del valor esperado poblacional.

Los valores poblacionales μ_x de *Rc*, *P*, *F1* y *FNR* tienen bajo desempeño respecto a las medias muestreadas μ_{ch} de *Baseline*. Esto sucede porque en esta categoría no se consideran problemas significativos en el escenario. Sin embargo, esta diferencia en *Rc*, *P*, *F1* y *FNR* indican una situación interesante respecto al comportamiento de SOM-CNN: cuando un video no tiene problemas significativos en el escenario, el modelado de fondo será modelado correctamente y los objetos dinámicos serán segmentados coherentemente. Aunado a esto, si existen problemas relacionados a Fondos dinámicos, bajos contrastes, sombras, reflejos, objetos dinámicos estacionarios, vibración de cámara, etc, el fondo será modelado correctamente, pero la coherencia en la segmentación de objetos dependerá de la severidad del problema teniendo un desempeño que va a caer en los valores de μ_x de *Rc*, *P*, *F1* y *FNR*.

Caracterización de SOM-CNN. Para entender cómo influyen las diversas situaciones del el escenario en el resultado de la segmentación, se van a definir a continuación una serie de puntos, la mayoría sustentados en la evaluación de SOM-CNN.

1). Si el video está en condiciones normales entonces el desempeño de SOM-CNN dará una segmentación de objetos dinámicos coherente.

2) Si existen objetos dinámicos estacionarios, SOM-CNN podrá segmentar el objeto incluso durante miles de cuadros. Esto sucede en videos como '*OF*', '*CH*' o '*LR*'.

3) Si existe un fondo dinámico que cubre la mayor parte del escenario, SOM-CNN removerá de manera exitosa el ruido pero existirán problemas de falsos negativos. Esto afectará directamente en los parámetros de *Rc*, *FNR* y *F1*. Se espera que el valor de las métricas varíe en el intervalo definido por μ_x en función de la severidad del ruido y del camuflaje entre el fondo y los objetos dinámicos.

4) Si existe un fondo dinámico que no cubre la mayor parte del escenario, es posible que SOM-CNN reduzca el ruido, pero no por completo ya que los valores de μ_d no son suficientes para aumentar las magnitudes de las sigmoidales de α, σ_β, a_1 y a_2. Esto afectará los parámetros de *P*, *FPR* y *F1*. Ejemplo de esto son los videos '*FL*' y '*FT*'.

5) Si la cámara tiene vibraciones, SOM-CNN las detectará como si existiera un Fondo dinámico muy severo, causando problemas de falsos negativos como se ve en los valores de *Rc* y *FNR* de μ_{ch} de *cameraJitter* de la Tabla 3.9. Este problema causa que las métricas se comporten en el sesgo no deseado de μ_x.

6) Cuando existe un objeto de fondo que cambia de lugar o es removido, SOM-CNN puede tardar decenas de cuadros en actualizar ese cambio. Esto afecta directamente en las métricas de *P, FPR, F1* y *PWC* como se ve en la Tabla 3.7 en los videos '*AX*', '*TS*' o '*WD*'. Este problema causa que las métricas *P, FPR, F1* y *PWC* se comporten en el sesgo no deseado de μ_x.

7) En escalas de grises y escenarios obscuros, SOM-CNN tiene un buen desempeño en todas las métricas. El promedio μ_{ch} esta dentro de los intervalos definidos en μ_x. Solo en casos donde el camuflaje no permite diferenciar entre el objeto dinámico y el fondo es cuando SOM-CNN tiene problemas.

8) Cuando existen reflejos y sombras, SOM-CNN logra detectarlos, pero no siempre se puede obtener un resultado satisfactorio en la segmentación de objetos dinámicos. Esto se puede observar en el video '*CB*'.

9). El camuflaje es un problema que en ocasiones causa errores en el desempeño de SOM-CNN. Ya que tal como se vio en los experimentos, si este problema se presenta, las métricas de *Rc, FNR, F1* tendrán resultados en el sesgo no deseado de los resultados de μ_x.

10) Aunque los cambios de iluminación graduales y repentinos no se analizan en *changedetection*, si se pudo observar en otras experimentaciones que SOM-CNN logra detectar exitosamente cuando estos problemas se presentan.

3.5.5 Estudio comparativo.

El análisis de los resultados de SOM-CNN con los videos de *changedetection* tiene como objetivo realizar un análisis comparativo de los métodos de segmentación presentes en la literatura y en la industria. Aunque no se realizó para este trabajo de tesis el procedimiento formal para ser publicado el método SOM-CNN en *changedetection.net*, si se realizó un estudio comparativo basándose en las herramientas que se ofrecen en dicha pagina. Estas herramientas son el paquete de software versión MATLAB® para obtener las métricas y las tablas con los resultados de diversos métodos. Estas tablas se van actualizando cuando un autor decide publicar sus resultados. Aunque múltiples autores han utilizado esta base de datos, en *changedetection.net*, solo aparecen los primeros 30, ya que quienes deciden participar en la comparación formal tienen la opción de publicar o no sus resultados.

Para esta tesis, la comparación puede ser de suma utilidad para conocer como esta posicionado SOM-CNN en el estado del arte de los métodos de segmentación de objetos dinámicos y conocer la contribución de SOM-CNN en el estado del arte. Por lo tanto, esta comparación se realizará bajo esta premisa.

El estudio comparativo de SOM-CNN se realizó en repetidas ocasiones desde Diciembre de 2013 utilizando siempre las tablas más recientes, pero en este trabajo de tesis se documenta la última comparación con las tablas actualizadas al 6 de abril 2014, las cuales están en el anexo 1 de la tesis. La comparación se realiza por categoría y luego para todas las categorías. En la Tabla 3.11 se muestran los resultados del estudio comparativo, los cuales representan el lugar en donde quedaría SOM-CNN posicionado contra la cantidad de métodos publicados.

Tabla 3.11 Resultados del estudio comparativo de SOM-CNN con los métodos participantes en *changedetection.net*.

Baseline	**cameraJitter**	**dynamicBackgraund**	**intObjectMotion**	**Shadow**	**Thermal**	**Lugar**
8/31	16/30	17/33	20/31	29/32	9/32	**17/29**

Como se puede observar en la Tabla 3.11, la categoría *baseline* es en la que SOM-CNN quedo mejor posicionado, *thermal* fue la segunda donde quedó mejor posicionado. Curiosamente, *cameraJitter* queda en tercer lugar a pesar de ser una categoría donde los resultados de SOM-CNN no son tan buenos. La categoría con el peor lugar fue *shadow* ya que SOM-CNN apenas supera a tres de los métodos publicados. Para entender mejor el posicionamiento de SOM-CNN, a continuación se realizará una comparación de este método con los mejores métodos de cada categoría y el mejor método de todas las categorías.

Baseline: En esta categoría el mejor método es SC-SOBS [128], el cual es una actualización del método SOBS que incluye una reducción en falsos positivos y una adaptabilidad de parámetros basándose en conocimiento previo del escenario. En la Tabla 3.12 se observan los resultados de SOM-CNN y SC-SOBS así como la diferencia entre ambos métodos. Se puede observar que las diferencias entre SC-SOBS y SOM-CNN no son significativas. Incluso, la diferencia en los parámetros entre los primeros 10 lugares parece más cuestión de suerte, ya que todos los valores tienen mucha similitud entre sí. Se puede apreciar que SOM-CNN tiene ligeramente mejores resultados en *Rc*, *FNR* y *PWC*. Esto quiere decir que SOM-CNN tiene menos falsos negativos que SC-SOBS. *PWC* indica que la relación falsos negativos/positivos mejora ligeramente en SOM-CNN.

Tabla 3.12 Resultados de SC-SOBC vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.9529	0.9970	0.0029	0.0470	0.4122	0.8937	0.9219
SC-SOBS	0.9327	0.998	0.0020	0.0673	0.3747	0.9341	0.9333
Diferencia	0.0202	-0.0010	0.0010	-0.0203	0.0375	-0.0400	-.01136

cameraJitter: De acuerdo a la información disponible en *changedetection.net*, en esta categoría los mejores resultados fueron obtenidos por el método anónimo de GPRMF. En la Tabla 3.13 se observan los resultados de SOM-CNN y GPRMF así como la diferencia entre ambos métodos. Se puede observar que entre GPRMF y SOM-CNN existen diferencias significativas en *R*, *FNR*, *P* y *F1*. Esta diferencia indica que SOM-CNN tiene un desempeño regular respecto a los métodos del estado del arte para detectar correctamente objetos dinámicos cuando se presentan vibraciones en la cámara. No obstante, SOM-CNN tiene ligeras mejorías en *Sp* y *FPR*.

Tabla 3.13 Resultados de GPRMF vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.4298	0.9958	0.0042	0.5701	2.4386	0.8069	0.5598
GPRMF	0.8159	0.9953	0.0047	0.1841	1.1062	0.9244	0.8596
Diferencia	-0.3860	0.0005	-0.0005	0.3861	1.3324	-0.1180	-0.2997

dynamicBackground: De acuerdo a la información disponible en *changedetection.net*, en esta categoría los mejores resultados fueron obtenidos por el método anónimo de DMB. En la Tabla 3.14 se observan los resultados de SOM-CNN y el método anónimo GPRMF así como la diferencia entre ambos métodos. Se puede observar que entre SOM-CNN y DMB existen diferencias significativas en *Rc* y *FNR*, aunque esta diferencia no es tan marcada cono en *cameraJitter*. Esta diferencia con respecto al mejor método en fondos dinámicos indica que SOM-CNN tiene un desempeño aceptable respecto a los métodos del estado del arte para detectar objetos dinámicos cuando se presentan estos problemas en el escenario. Para el resto de las métricas existen diferencias poco significativas, por lo que SOM-CNN tiene un desempeño aceptable en problemas de Fondos dinámicos.

Tabla 3.14 Resultados de DMB vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.6224	0.9977	0.0023	0.3776	0.502105	0.7289	0.66508
DMB	0.9155	0.9987	0.0013	0.0845	0.2282	0.7877	0.8262
Diferencia	-0.2930	-0.0010	0.0010	0.2931	0.273905	-0.059	-0.16112

intermittentObjectMotion: En esta categoría el mejor método fue SGMM-SOD [129]; es un método diseñado para detectar objetos dinámicos en escenarios con multitudes de personas, el cual es robusto para detectar objetos que son removidos del fondo y objetos dinámicos estacionarios. En la Tabla 3.15 se observan los resultados de SOM-CNN y SGMM-SOD así como la diferencia entre ambos métodos. Se puede observar que entre SOM-CNN y SGMM-SOD existen diferencias significativas en *P*, *F1* y *PWC*. Estas diferencias indican que SOM-CNN generó muchos falsos positivos con algunos videos de esta categoría. Como se mencionó anteriormente, estos falsos positivos se deben a los videos donde objetos que eran de fondo fueron removidos en varias decenas de cuadros. El resto de las métricas no tienen diferencias significativas, por lo cual el desempeño de SOM-CNN respecto a los métodos del estado del arte es aceptable para los videos de esta categoría.

shadow: De acuerdo a la información disponible en *changedetection.net*, en esta categoría los mejores resultados fueron obtenidos por el método anónimo de GPRMF. En la Tabla 3.16 se observan los resultados de SOM-CNN y GPRMF así como la diferencia entre

ambos métodos. Se puede notar que en esta categoría existen diferencias entre GPRMF y SOM-CNN en las métricas de *P*, *Rc* y *F1*. Estas diferencias indican que GPRMF tuvo mejor desempeño que SOM-CNN para detectar coherentemente los objetos dinámicos. Esta diferencia indica que el desempeño de SOM-CNN en esta categoría es aceptable respecto a los métodos del estado del arte.

Tabla 3.15 Resultados de SGMM-SOD vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.6973	0.9560	0.0440	0.3027	6.3036	0.4893	0.5027
SGMM-SOD	0.7363	0.9909	0.0091	0.2637	2.5238	0.8141	0.7151
Diferencia	-0.0389	-0.0349	0.0349	0.0389	3.7798	-0.3250	-0.2124

Tabla 3.16 Resultados de GPRMF vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.7420991	0.98676	0.01023	0.257901	2.103831	0.6515	0.68073
GPRMF	0.9253	0.9922	0.0078	0.0747	1.0712	0.8889	0.8671
Diferencia	-0.183201	-0.0054	0.00243	0.183201	1.032631	-0.237	-0.18637

thermal: En esta categoría el mejor método fue STLBP [130]. Este método modela el fondo con base a información espacio temporal e histogramas de textura. En la Tabla 3.17 se observan los resultados de SOM-CNN y STLBP así como la diferencia entre ambos métodos. Se puede observar que las diferencias entre STLBP y SOM-CNN son poco significativas excepto en la métrica de *P*. Esto indica que SOM-CNN arrojó más falsos positivos STLGP en la detección del objeto dinámico. Sin embargo, SOM-CNN tuvo mejor desempeño que STLGP en *PWC*, lo cual indica que la detección de falsos positivos/negativos fue mejor en SOM-CNN que en STLBP. Esto indica que el desempeño de SOM-CNN detectando los objetos dinámicos fue tan bueno como STLBP en esta categoría, pero SOM-CNN generó más ruido.

Tabla 3.17 Resultados de STLBP vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.7169	0.9938	0.0062	0.2831	1.4527	0.8020	0.7514
STLBP	0.7231	0.9993	0.0007	0.2769	2.9861	0.9798	0.8067
Diferencia	-0.0066	-0.0055	0.0055	0.0062	-1.5334	-0.1770	-0.0553

Todos los videos: El método que tuvo el primer lugar en todas las categorías hasta el 6 de abril de 2014 en el *DATASET 2012* fue *Spectral-360*. Este método está documentado en la patente de [131]. De acuerdo a http://wordpress-jodoin.dmi.usherb.ca/method/66/, este método tiene una serie de parámetros que se dejaron constantes para la participación en *changedetection*. Además, se implementó en visual C++ y su velocidad de procesamiento

es de 12 fps en videos con resolución de 320x240. En la Tabla 3.18 se observan los resultados de SOM-CNN y STLBP así como la diferencia entre ambos métodos.

Tabla 3.18 Resultados de *Spectral-360* vs SOM-CNN.

Método	Rc	Sp	FPR	FNR	PWC	P	F1
SOM-CNN	0.69313	0.9865	0.0129	0.3068	2.3266	0.7107	0.67017
Spectral-360	0.7770	0.9920	0.0080	0.2230	1.8516	0.8461	0.7770
Diferencia	-0.0838	-0.0055	0.0049	0.0839	0.4749	-0.1350	-0.1068

De acuerdo a los resultados de la Tabla 3.18, *Spectral-360* y SOM-CNN no tienen muchas diferencias significativas en *Rc*, *Sp*, *FPR*, *FNR* y *PWC*. Esto implica que ambos métodos modelan el fondo y detectan los objetos dinámicos de manera similar. No obstante, existen diferencias significativas en las métricas de *P* y *F1*. Esto indica que SOM-CNN generó más ruido que *Spectral-360*. De acuerdo a las tablas de las categorías en *changedetection.net*, el método *Spectral-360* tuvo su mejor desempeño en la categoría de *shadow*, mientras que en el resto de las categorías queda en los primeros siete lugares. La diferencia entre este y el resto de los métodos que están en los primeros lugares es muy poca, incluso con base a las métricas documentadas en las tablas, se podría ver que todos los métodos que se encuentran en los primeros lugares funcionan de manera similar.

Este estudio comparativo reafirma lo que se ha concluido anteriormente; cuando SOM-CNN detecta fondos dinámicos, se generan falsos negativos por el comportamiento de parámetros que los mejores métodos de detección de movimiento, teniendo un desempeño aceptable. Cuando un objeto es removido del fondo, SOM-CNN genera más ruido que los mejores métodos de detección de movimiento, teniendo un desempeño regular. No obstante, para eliminar ruido en fondos dinámicos, escenarios con muy bajos contrastes, objetos dinámicos estacionarios, condiciones normales, SOM-CNN tiene resultados tan buenos como los mejores métodos de detección de movimiento. Como no existen diferencias muy significativas entre SOM-CNN y los métodos que están en los primeros lugares, se puede notar que este método tiene un desempeño equiparable con los mejores métodos que participan en la comparación de *changedetection.net*.

3.6 Conclusiones

En este capítulo se presentó de manera extensiva el desarrollo de dos RNA para análisis de video, un método de segmentación de objetos dinámicos y su evaluación de desempeño. Las RNA propuestas en este capítulo son RESOM y NTCNN, el método de

segmentación es SOM-CNN y la evaluación del desempeño se realizó con diversos videos muy utilizados en la literatura. Este fue un capítulo muy extenso, por lo que esta sección de conclusiones se divide en tres subsecciones que contienen las conclusiones de las tres principales contribuciones propuestas en este capítulo: RESOM, de NTCNN y del método de segmentación SOM-CNN.

3.6.1 Conclusiones de RESOM

SOM Retinotópica (RESOM) es una RNA inspirada en el mecanismo de aprendizaje de la SOMRM, la cual es un modelo neurocomputacional basado en Auto-organización de pesos que simula los mecanismos de aprendizaje para la organización cortical las capas más bajas de la corteza visual. RESOM es una red con la misma interacción Retinotópica que SOMRM, y tiene una competición entre neuronas así como un aprendizaje de los pesos se basado en la regla Hebbiana. RESOM utiliza dos parámetros de aprendizaje (α y $\beta(m)^z$) para modelar la capacidad de aprendizaje de la red, y por lo tanto definir estados de aprendizaje para seleccionar la información que se desea conservar de una secuencia de video. Esta flexibilidad dada por estos parámetros se puede utilizar no solo para modelado de fondo, también se puede aplicar para múltiples aplicaciones relacionadas con análisis de video, como diseño de algoritmos de atención y alerta visual, segmentación de color en video, o cualquier aplicación donde se puedan generar múltiples mapeos sensibles a ciertas características del escenario. Otro aspecto interesante de RESOM es su rapidez en el procesamiento, ya que en MATLAB® tiene velocidades de procesamiento de 165 fps en videos con resoluciones de 128x160 y 30 fps en videos con resoluciones de 240x320. Sin embargo, en pruebas realizadas en C++, se pudo observar que RESOM tiene velocidades de procesamiento de 500 fps con videos de 128x160 y 96 fps en videos de 240x320.

Con base a RESOM, se diseñó un método de modelado de fondo denominado SOM Retinotópica Dinámica (DR-SOM), en el que α y $\beta(m)^z$ generan diferentes estados de aprendizaje donde RESOM puede aprender solo la información de los objetos de fondo o el escenario completo. Esto permitió a RESOM generar un modelado de fondo que se adapta fácilmente a cualquier cambio que exista en el escenario. Esto se pudo evidenciar en los experimentos de SOM-CNN, el cual se adapta fácilmente a segmentación de objetos que permanecen estacionarios por varios cuadros como a cambios de iluminación.

Por la capacidad de RESOM para adaptarse a cualquier aplicación relacionada con análisis de escenario, sus velocidades de procesamiento y su aplicación exitosa para modelar el fondo en DR-SOM, se puede concluir que RESOM es un modelo bioinspirado V1 que puede ser muy exitosa en diferentes aplicaciones relacionadas al análisis de video.

Respecto a trabajos futuros relacionados con RESOM pueden existir varias líneas de investigación. Una de ellas es el diseño de una red RESOM donde la competición que parte de la ecuación 3.4 tenga una función de activación *f(.)* que sea no lineal, así como modificaciones para hacer una regla Hebbiana retroalimentada. Como se mencionó en el capítulo II, esta competición simula la respuesta de V1 y es la responsable de la selectividad de la información aprendida y la regla Hebbiana es responsable de organizar esta capa cortical. Estos cambios permitirían tener un mayor control en la selectividad de la neurona ganadora y generar vectores de pesos retinotópicos que mapeen ciertas características del escenario.

3.6.2 Conclusiones de NTCNN

La Red Neuronal Celular de Binarización por Vecindario (NTCNN) es una RNA para segmentación que mapea el valor de los pixeles a dos estados. Esta es una Red Neuronal Celular de Binarización (TCNN), pero con una modificación inspirada en el mecanismo de las superficies monoculares de LAMINART dado por la ecuación 2.4. La NTCNN es en realidad una TCNN que selecciona un umbral a cada pixel con base a un parámetro global *z* y dos parámetros locales a_1 y a_2. Estos parámetros hacen de NTCNN una red muy exitosa para la binarización de cualquier información que se encuentre a escalas de grises ya que este modelo utiliza la información del vecindario en las primeras iteraciones para generar los umbrales en cada pixel y en las ultimas iteraciones realiza la binarización.

En SOM-CNN la NTCNN fue una forma muy exitosa de binarización ya que permitió reducir el ruido afectando mínimamente la segmentación coherente de objetos en escenarios con problemas de fondos dinámicos, ruido y videos con movimiento de cámara. La velocidad de procesamiento de la NTCNN con seis iteraciones fue de 300 fps en MATLAB®, 100 fps en C++. No obstante, una versión de la NTCNN implementada en GPU *PNY GEOFORCE GTX 550 Ti* generó velocidades de procesamiento cercanas a 1000

fps. Por lo tanto, la implementación en GPU genera velocidades similares a cualquier algoritmo de binarización que no es iterativo y no utiliza información de vecindario.

Con base a las velocidades de procesamiento de GPU y la flexibilidad que NTCNN le brinda a SOM-CNN para segmentar $I_e(x,y)^t$ en situaciones muy diversas, se pude concluir que NTCNN es un modelo muy robusto que permite diseñar diferentes estrategias para binarizar información en escalas de grises. Como trabajo futuro se puede definir una NTCNN con una plantilla de pesos $A(i_N,j_N)$ no simétrica cuyos coeficientes se adapten de acuerdo a la información que dicho modelo va a procesar.

3.6.3 Conclusiones de SOM-CNN

SOM-CNN es un método de segmentación de objetos dinámicos compuesto de tres módulos: Modelado de fondo basado en DR-SOM, Binarización por NTCNN y el Tercer módulo para ajuste automático de parámetros.

El módulo DR-SOM permite a SOM-CNN definir los estados de aprendizaje necesarios para modelar el fondo de acuerdo a las condiciones del escenario y los cambios que este va teniendo. Esto se debe a que SOM-CNN puede modelar el aprendizaje de todas las neuronas manejando el parámetro α, puede modelar el aprendizaje de cada neurona con $\beta(m)^z$. Este aprendizaje permite a SOM-CNN ser robusto a múltiples condiciones en un escenario que van desde cambios de iluminación repentinos hasta no anexar al fondo objetos dinámicos que permanecen estacionarios por cientos de cuadros.

El módulo NTCNN permite a SOM-CNN generar una segmentación flexible con a_1 y a_2 para adaptarse a las condiciones del escenario, de manera que el ruido en $I_e(x,y)^t$ no afecte a la segmentación de objetos. En caso de no existir ruido, la NTCNN se enfocará a segmentar a mayor detalle posible los objetos dinámicos.

El tercer modulo maneja los parámetros α, $\beta(m)^z$, a_1 y a_2. Este módulo permite que SOM-CNN sea robusto a cambios de iluminación graduales y repentinos, fondos dinámicos, *bootstrapping*, objetos dinámicos que permanecen estacionarios, escenarios obscuros y con bajo contraste. De acuerdo a las pruebas de desempeño realizadas en la sección 3.4, SOM-CNN tiene el mejor desempeño que el resto de los métodos en la literatura para detectar cambios de iluminación y adaptarse rápidamente sin sacrificar la segmentación de objetos durante la transición. Además, de acuerdo a las pruebas de

desempeño y validación de la sección 3.4 y 3.5, SOM-CNN tiene un desempeño equiparable con los métodos del estado del arte en escenarios con fondos dinámicos, objetos dinámicos estacionarios, escenarios obscuros y con bajos contrastes. Por otro lado, SOM-CNN tiene problemas con objetos de fondo que son removidos de este, camuflaje y falsos negativos cuando existen fondos dinámicos.

Con base a diversos experimentos realizados donde los parámetros α, $\beta(m)^z$, a_1 y a_2 son ajustados manualmente, los problemas de objetos de fondo que son removidos y de falsos negativos en fondos dinámicos se pueden resolver ya que se originan de la manera en como el Tercer módulo y no a limitantes en el funcionamiento de la RESOM o la NTCNN.

Los tiempos de procesamiento de SOM-CNN mostraron que es un método plausible para aplicaciones de tiempo real ya que se procesa a 50 fps en MATLAB® con videos cuya resolución espacial es 128x160. En una versión de SOM-CNN donde los parámetros son ajustados manualmente los tiempos de procesamiento fueron de 77 fps en MATLAB® con videos con resolución espacial es 128x160. Esta misma versión fue implementada en C++/GPU en una *Workstation Dell Precision* con procesador Intel *Xeon* y con un GPU *PNY GEOFORCE GTX 550 Ti* y los tiempos de procesamiento fueron 200 fps para videos con videos con resolución espacial es 128x160 y 45 fps en videos con resolución espacial de 240x320. Si se toma en cuenta que el método *Spectral-360* reporta un tiempo de procesamiento de 12 fps con resolución espacial es 320x240, se puede entonces concluir que SOM-CNN es uno de los métodos más rápidos de segmentación de objetos dinámicos.

En conclusión, SOM-CNN es un método adaptivo diseñado para aplicaciones en tiempo real que es robusto a cambios de iluminación graduales y repentinos, fondos dinámicos, escenarios con bajos contrastes, *bootstrapping* y objetos dinámicos que permanecen estacionarios. Las principales contribuciones de SOM-CNN en la literatura radican en el tiempo de procesamiento que tiene el método y la capacidad de este para adaptarse a las diferentes condiciones del escenario que van desde cambios de iluminación repentinos hasta mantener la segmentación de objetos que permanecen cientos de cuadros sin moverse.

Como trabajo futuro se pretende rediseñar el tercer módulo basándose en algún modelo basado en RNA cuyo comportamiento se inspire en el módulo MT de la corteza

visual. Esto permitiría eliminar los múltiples umbrales, además permitiría ajustar cada parámetro de DR-SOM y NTCNN en forma independiente y no todos como sucede en la actual de SOM-CNN.

VI. MÉTODOS DE SEGMENTACIÓN DE OBJETOS ESTÁTICOS BASADO EN ENFOQUE HÍBRIDO

En el capítulo anterior se documentó el desarrollo de SOM-CNN, un método de segmentación de objetos dinámicos basado en las redes neuronales RESOM y NTCNN. Aunque la segmentación de objetos es satisfactoria, la información del fondo de cada cuadro $I(x,y,z)^t$ no se utiliza, y tal como se mencionó en el capítulo I, una hipótesis en este trabajo de tesis es que la información del fondo es tan útil como la información de los objetos de movimiento ya que se puede realizar una mejor interpretación del contenido del escenario. Por ejemplo, en ocasiones los objetos dinámicos estacionarios pueden estar fijos durante ventanas de tiempo muy grandes y por más robusto que sea el método de segmentación de objetos dinámicos, estos en algún momento se van a integrar al fondo. Otro ejemplo es en predicción de movimiento, ya que al segmentar los objetos estáticos, es posible encontrar de forma más sencilla las posibles oclusiones.

En esta tesis, objeto estático se definió como los objetos que componen el fondo. Por lo tanto, segmentación de objetos estáticos se refiere a dividir el resultado del modelado de fondo en un conjunto de regiones que representen de forma coherente cada uno de los objetos que componen el fondo de una secuencia de video. Sin embargo, en la literatura no hay información referente a la segmentación de objetos estáticos, ya que la mayor parte del trabajo relacionado con análisis de video se enfoca a segmentación y seguimiento de objetos en movimiento. Entre los trabajos que resuelven problemas similares a segmentar objetos estáticos, en [19], un método basado en Wavelet y SOM realiza una segmentación de color en secuencias de video, en [20] una SOM jerárquica realiza una separación de regiones que representan los objetos de una secuencia de video para posteriormente segmentar las regiones que se van moviendo entre cuadros sucesivos. Existen otros trabajos como en **[**18**,**132**]** donde un método con un enfoque neuroinspirado realiza una separación de regiones en un escenario de una secuencia de video para encontrar los objetos más sobresalientes o con cierta pregnancia.

En **[**133**]** se propone otro método de segmentación, donde los objetos de fondo son segmentados mediante agrupamiento por *kmeans* y CNN. Dicho método se basa en encontrar el modelado de fondo en el espacio HSV, agrupar por *kmeans* con cuatro clases y

generar el resultado de la segmentación con CNN. En este trabajo de tesis, se complementó el método de segmentación de objetos estáticos propuesto en **[**133**]** con un análisis para encontrar la cantidad de regiones en que se debe separar cada cuadro. Este complemento consiste en un análisis de color del escenario de la secuencia de video para encontrar la cantidad de centroides que *kmeans* debe obtener. Al realizar esto, el método de segmentación debe tener ciertas modificaciones, ya que la cantidad de regiones pueden cambiar durante la secuencia de video si se modifican las condiciones del escenario debido a nuevos objetos que se anexan al fondo, cambios de iluminación, fondos dinámicos, etc. La principal contribución de un método de segmentación estática es el análisis coherente del escenario en una secuencia de video, ya que en tareas de segmentación del escenario contempla solo el análisis espacial.

Es decir, en la literatura es común encontrar diferentes trabajos relacionados a segmentación de imágenes. Se pueden encontrar un sinfín de métodos para segmentar color, textura, bordes y objetos utilizando cientos de modelos basados en procesamiento clásico de imágenes, RNA, modelos probabilísticos, etc **[**6**,**134**,**135**,**9**]**. No obstante, la mayor parte de estos métodos no son plausibles para segmentación de objetos estáticos debido a que fueron diseñados y probados con imágenes y no con secuencias de video. Una imagen contiene información de un escenario en un instante determinado, mientras que una secuencia de video tiene información de un escenario en un intervalo de tiempo. Por lo tanto, la información contenida en una imagen es por definición invariante en el tiempo, mientras que en un video por definición la información es variante en el tiempo. Aunque existen muchos métodos que son exitosos para segmentar en imágenes, la mayoría de estos no consideran que la información espacial sea también variante en el tiempo.

Sin embargo, en secuencias de video se deben tener dos tipos de coherencias en los resultados de segmentación: espacio y tiempo. La coherencia en el espacio se refiere a que la partición del cuadro en regiones representa correctamente los objetos o sus propiedades presentes en el escenario. La coherencia en el tiempo se refiere a que la partición de regiones sigue de manera correcta los cambios que el escenario va presentando a lo largo de la secuencia de video. Dado que los métodos en la literatura para segmentación de imágenes no garantizan la coherencia en el tiempo, esta puede ser una aportación de los métodos de segmentación de objetos estáticos; proponer un método que realice una

segmentación coherente en el espacio y en el tiempo. Además, un aspecto adicional que se debe considerar en un método de segmentación de objetos estáticos es que los tiempos de procesamiento tienen que ser lo suficientemente rápidos para analizar cuadros de manera consecutiva, mientras que muchos métodos de segmentación de imágenes no consideran este aspecto.

Por lo tanto, dado que no se han propuesto trabajos relacionados a segmentar un escenario aquirido en una secuencia de video, se consideró en esta tesis que desarrollar un método que logre la coherencia en espacio y tiempo en los resultados es una aportación en el mejoramiento de algoritmos de segmentación. Antes de plantear un esquema de segmentación de objetos estáticos neuroinspirados, se complementó el método propuesto en **[**133**]** para posteriormente compararlo con un método de segmentación neuroinspirado. En la Figura 4.1 se puede observar un diagrama del método de segmentación completo, el cual se denomina KCNN. Los módulos en color gris de la Figura 4.1 son propuestos en **[**133**]**; mientras que los módulos en azul de la Figura 4.1, son propuestos en este trabajo.

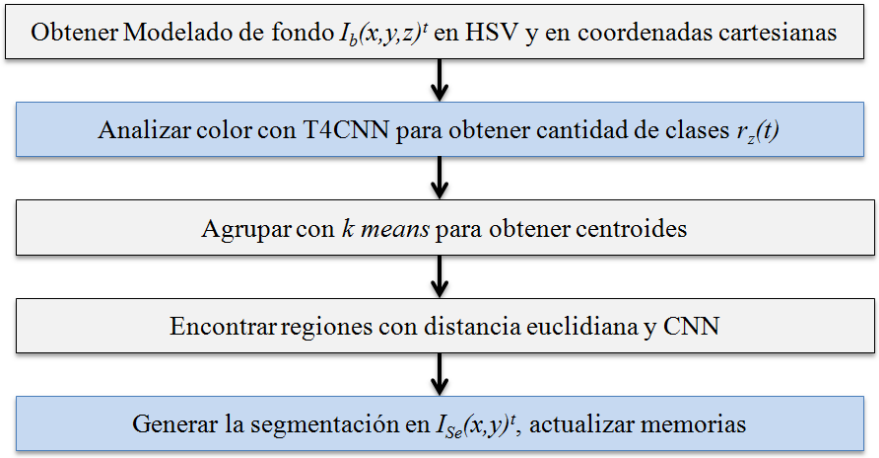


Figura 4.1. Módulos del método de segmentación KCNN.

La entrada de este método es un cuadro de una secuencia de video que se mapea a HSV y es representado en coordenadas cartesianas. El primer módulo se refiere a obtener el modelado de fondo, lo cual puede ser con el método propuesto en [133] o por DR-SOM. Luego, se encuentra la cantidad de regiones en que se va a realizar la segmentación, para lo cual en este trabajo se propone la T4CNN, una red neuronal celular que analiza una imagen para definir cómo se comportan los niveles de grises. Se aplica una T4CNN en cada uno de los componentes de color en HSV para encontrar la cantidad de clases que el siguiente

módulo de *kmeans* debe utilizar para encontrar el número $r_z(t)$ de centroides para cada cuadro *t*. Después, el siguiente módulo genera las regiones con la distancia euclidiana entre los centroides y la información contenida en los pixeles. Con la CNN se unen las regiones para generar un cuadro $Is_e(x,y)^t$ que contiene la segmentación de objetos estáticos. Finalmente, el resultado de $Is_e(x,y)^t$ y los centroides son almacenados en una memoria de datos e imágenes que se utilizan para dar coherencia al seguimiento de regiones a lo largo de la secuencia de video. Dado que los módulos de modelado de fondo, *kmeans* y generación de regiones ya se documentaron en [133], en el resto de este capítulo se documentan los módulos de T4CNN y memorias de imágenes. Finalmente se reportan resultados y conclusiones preliminares del método de segmentación.

4.1. Módulo de análisis de color con CNN

La DTCNN ha sido a una RNA útil principalmente en tareas como segmentación, detección de bordes, esquinas o para procesamiento visual en etapas iniciales [136]. En la sección 3.3, se mostró el desarrollo de una CNN para binarización con una plantilla que en las primeras iteraciones elimina ruido y el resto de las iteraciones realiza una umbralización. Las plantillas que se utilizaron para la NTCNN se inspiraron a partir de la TCNN y también existe cierta inspiración de plantillas de pesos para análisis de color utilizadas en **[**137**,**138**].** La diferencia entre estos métodos y la NTCNN está en que en el caso de [137] se utiliza una tercer plantilla de pesos para relacionar en una sola CNN los tres canales de color. En el caso de **[**138**],** se utiliza una plantilla de pesos *A* dada por:

$$A = \begin{bmatrix} 0 & 0.1 & 0 \\ 0.1 & 0.9 & 0.1 \\ 0 & 0.1 & 0 \end{bmatrix} \tag{4.1}$$

Las plantillas *B* y Z_c son cero, como en la TCNN. En ambos casos, la segmentación de color se reporta en forma visual con ejemplos; es decir, no existe una medición del desempeño. Sin embargo, a partir de estos trabajos se diseñó en este trabajo de tesis una estrategia basada en CNN para completar un procesamiento de segmentación de objetos estáticos y solucionar la limitante de las 4 clases en [133]**.** Para esto, se procedió a realizar un análisis para encontrar la distribución de color en $I_b(x,y,z)^t$ basándose en los modelos de

segmentación con la CNN. Para lo cual, se realizó una modificación de la TCNN apoyandose en los trabajos mencionados anteriormente, las plantillas resultantes fueron:

$$A=\begin{bmatrix} 0 & 0.1 & 0 \\ 0.1 & 1 & 0.1 \\ 0 & 0.1 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z_c = 0 \tag{4.2}$$

Las condiciones iniciales de la red son:

Entrada: Imagen *I(i,j)* de ceros.
Estado inicial: $\chi(i,j,0)^z=I_{ec}(i,j)^z$, donde $I_{ec}(i,j)^z=1-2I_b(x,y,z)^t$. Para esta CNN, $x=i$, $y=j$. Se va a aplicar una instancia de la T4CNN para cada componente de color de $I_b(x,y,z)^t$.

Sustituyendo las plantillas de pesos de la ecuación 4.2 en la ecuación 3.27 (ecuación de estado de la CNN), se obtiene:

$$\chi(i,j)^{n_c+1,z} = f\left[\chi(i,j)^{n_c,z}\right] + 0.1 \sum_{(i_N,j_N)\in Nr4} f\left[\chi(i_N,j_N)^{n_c,z}\right] \tag{4.3}$$

N_{r4} se refiere a los vecinos 4 de cada elemento *(i,j)*. La ecuación 3.28 que representa la salida $y_c(i,j)^z=f[\chi(i,j)^z]$ tiene el siguiente resultado con $N_c=10$ cuando $n_c\rightarrow\infty$:

$$y_c(i,j)^z=-1, \text{ si } I_{ec}(i,j)^z > f[\chi(i_N,j_N)^z].$$
$$y_c(i,j)^z=1, \text{ si } I_{ec}(i,j)^z \leq f[\chi(i_N,j_N)^z].$$

Luego, esta se mapea de la siguiente forma:

$$I_{bc}(x,y,z)^t = \left(1-yc(i,j)^{10,z}\right)/2 \tag{4.4}$$

Para cada canal $z=\{H,S,V\}$ de color se aplica este modelo. Este algoritmo se le denomina en la tesis Red Neuronal Celular de umbralización con vecinos (T4CNN).

4.1.1 Reglas para la obtención de la cantidad de regiones en el espacio de color HSV

El método que obtiene la cantidad de regiones estáticas, utiliza como entrada $I_b(x,y,z)^t$, pero descompuesta en una serie de componentes que en conjunto forman un arreglo de color que se denomina H^2SV, el cual tiene la información de matiz (*H)*, saturación (*S*), valor *(V)* y una modificación al componente de matiz denomiado H_2 dado por:

$$\begin{aligned} & I_b\left(x,y,H_2\right) = I_b\left(x,y,H\right) + \pi/2 \\ & si \quad I_b\left(x,y,H\right) > 2\pi \\ & \quad I_b\left(x,y,H_2\right) = Ib\left(x,y,H\right) - 2\pi \end{aligned} \tag{4.5}$$

Esta modificación de matiz H_2 contiene los valores de matiz rotados en un cuadrante ($\pi/2$), tal como se muestra en la Figura 4.2a. Por lo tanto, H^2SV tiene las componentes H, H_2, S y V. La componente H_2 se utiliza junto con H en pasos posteriores para conocer en que cuadrantes se localiza el color de $I_b(x,y,z)^t$. De esta manera, el método de procesamiento con la T4CNN consiste en lo siguiente:

1. Se mapea $I_b(x,y,z)^t$ al arreglo H^2SV.
2. Los componentes de color $I_b(x,y,H)$, $I_b(x,y,H_2)$, $I_b(x,y,S)$, $I_b(x,y,V)$ se utilizan para el estado inicial $\chi(i,j,0)^z=I_{ec}(i,j)^z$, donde $z=\{H,H_2,S,V\}$.
3. Las T4CNN se procesan con $N_c=10$.
4. Se calcula la salida de las T4CNN con la ecuación 4.4 y se obtiene lo siguiente:

$$\begin{aligned} & I_b(x,y,H)^t : T4CNN_H \Rightarrow I_{bc}(x,y,H)^t \\ & I_b(x,y,H_2)^t : T4CNN_{H_2} \Rightarrow I_{bc}(x,y,H_2)^t \\ & I_b(x,y,S)^t : T4CNN_S \Rightarrow I_{bc}(x,y,S)^t \\ & I_b(x,y,V)^t : T4CNN_V \Rightarrow I_{bc}(x,y,V)^t \end{aligned} \tag{4.6}$$

En la Figura 4.2a se puede observar la información resultante obtenida del arreglo de color H^2SV, donde se ve el espacio de matiz (H), matiz rotado (H_2), saturación (S) y valor (V). Mediante estos componentes, la información de cada pixel en $I_b(x,y,z)^t$ se puede describir de la siguiente manera:

- Si un pixel $I_b(x_1,y_1,z)$ tiene matiz rojo o amarillo, entonces, $I_{bc}(x,y,H)=0$ e $I_{bc}(x,y,H_2)=0$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene matiz verde-cyan, entonces, $I_{bc}(x,y,H)=0$ e $I_{bc}(x,y,H_2)=1$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene matiz cyan-azul, entonces, $I_{bc}(x,y,H)=1$ e $I_{bc}(x,y,H_2)=0$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene matiz azul-magenta, entonces, $I_{bc}(x,y,H)=1$ e $I_{bc}(x,y,H_2)=1$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene baja saturación $I_{bc}(x,y,S)=0$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene alta saturación $I_{bc}(x,y,S)=1$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene bajo luminancia $I_{bc}(x,y,V)=0$.
- Si un pixel $I_b(x_1,y_1,z)$ tiene alto luminancia $I_{bc}(x,y,V)=1$.

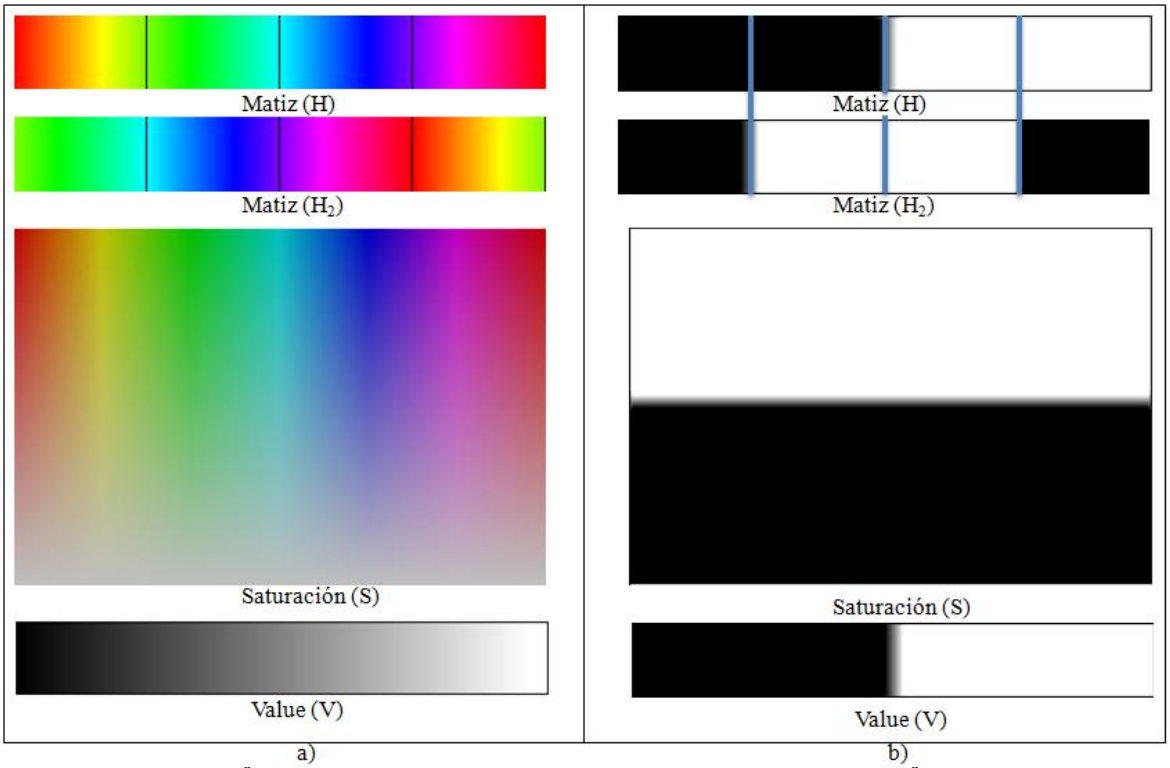


Figura 4.2. Espacio H^2SV y salida de la T4CNN. a) Respuesta de las componentes H^2SV en $I_b(x,y,z)^t$, donde $z=\{H,H_2,S,V\}$. b) $I_{bc}(x,y,z)^t$ correspondientes a a).

Un aspecto interesante que se puede observar en la Figura 4.2b es que la información de color se puede describir con base a los componentes de H y H_2, ya que se puede indicar a cual cuadrante de matiz ($[0,\pi/2)$, $[\pi/2,\pi)$, $[\pi, 3\pi/2)$, $[3\pi/2, 2\pi)$) pertenece la información de cada pixel. De la misma manera, la información de S y V se puede clasificar para cada pixel en S baja o alta y V bajo o alto.

Basándose en los puntos anteriores que analizan la información de color de cada pixel, se asume que en una imagen la distribución de color se puede medir de forma similar. En este caso, para estudiar la información de color de los pixeles de la imagen se puede realizar un análisis estadístico con un histograma de $I_{bc}(x,y,z)$, denominado $h_{bc}(i_L,z)$, donde $z=\{H,H_2,S,V\}$, i_L son los niveles de grises. Dado que $I_{bc}(x,y,z)$, tiene resultados casi binarizados, entonces los valores que interesan del histograma están en $i_L=\{0,1\}$. Este conjunto de histogramas se utilizará para definir la cantidad $r_z(t)$ de clases que el *kmeans* necesita para la segmentación en cada cuadro, de manera que los histogramas $h_{bc}(i_L,H)$ y $h_{bc}(i_L,H_2)$ definirán la cantidad de clases r_H que debe aportar el componente de matiz, el

histograma $h_{bc}(i_L,S)$ define la cantidad de clases r_S que debe aportar la información de saturación y el histograma $h_{bc}(i_L,V)$ define la cantidad de clases r_V que debe aportar la información de valor. En la Figura 4.3 se ilustran algunos ejemplos donde se tiene una imagen y los histogramas $h_{bc}(i_L,z)$. Como se observa en las Figura 4.3, los histogramas están normalizados con respecto a la mayor frecuencia de ocurrencia del histograma para manejar intervalos de [0,1] y no rangos variables.

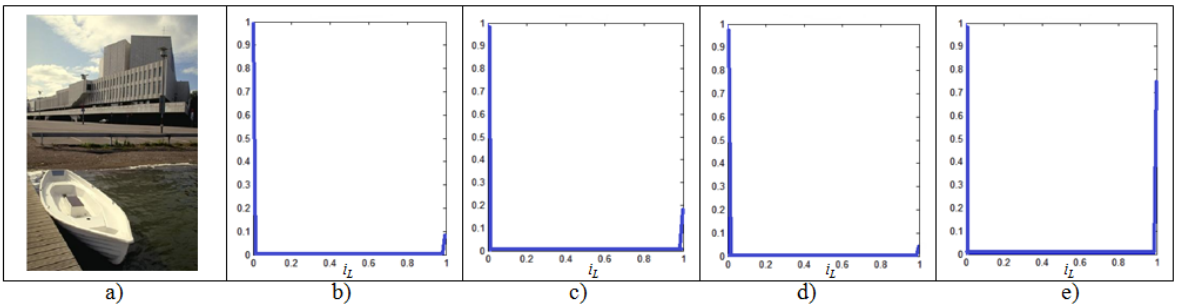


Figura 4.3. Histogramas. a). Imagen original Test #78004. b). $h_{bc}(i_L,H)$. c). $h_{bc}(i_L,H_2)$. d). $h_{bc}(i_L,S)$. e). $h_{bc}(i_L,V)$.

La componente de H es una información expresada como un hexácono donde cada vértice representa el nivel de matiz más puro de rojo, amarillo, verde, cyan, azul y magenta **[**121**,**122**,**139**].** Basado en esto, se asume que la componente de H debe generar hasta seis regiones dependiendo de la cantidad de matices que tenga la escena. Dado que la información de H y H_2 puede indicar en qué cuadrante se encuentra la información de matiz de un pixel, se puede asumir que los histogramas de $h_{bc}(i_L,H)$ y $h_{bc}(i_L,H_2)$ pueden indicar en cuál o cuáles cuadrantes se encuentran agrupados todos los niveles de matiz de los pixeles que contienen la escena. Si ΔH es un intervalo dentro de H y $\Delta H=\pi/2$, se pueden cubrir de uno a dos niveles de matiz. Si $\Delta H=\pi$, entonces se cubren tres niveles de matiz puro ya que cada matiz tiene un tamaño de $\Delta H=\pi/3$. Basado en esto, se diseñaron las siguientes reglas:

1. Si $h_{bc}(0,H)>0$ & $h_{bc}(1,H)=0$ & $h_{bc}(0,H_2)>0$ & $h_{bc}(1,H_2)=0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría anaranjados y amarillos, $r_H = 2$.
2. Si $h_{bc}(0,H)>0$ & $h_{bc}(1,H)=0$ & $h_{bc}(0,H_2)=0$ & $h_{bc}(1,H_2)>0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría verdes y cyan, $r_H = 2$.
3. Si $h_{bc}(0,H)=0$ & $h_{bc}(1,H)>0$ & $h_{bc}(0,H_2)=0$ & $h_{bc}(1,H_2)>0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría azules y cyan, $r_H = 2$.
4. Si $h_{bc}(0,H)=0$ & $h_{bc}(1,H)>0$ & $h_{bc}(0,H_2)>0$ & $h_{bc}(1,H_2)=0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría magenta y rojos, $r_H = 2$.

5. Si $h_{bc}(0,H)=0$ & $h_{bc}(1,H)>0$ & $h_{bc}(0,H_2)>0$ & $h_{bc}(1,H_2)>0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría cyan, azul, magenta, $r_H = 3$.
6. Si $h_{bc}(0,H)>0$ & $h_{bc}(1,H)=0$& $h_{bc}(0,H_2)>0$ & $h_{bc}(1,H_2)>0$, entonces, $I_b(x,y,z)^t$, tiene pixeles en su mayoría anaranjado, amarillo y verde $r_H = 3$.
7. Si $h_{bc}(0,H)>0$ & $h_{bc}(1,H)>0$ & $h_{bc}(0,H_2)>0$ & $h_{bc}(1,H_2)>0$, entonces, $I_b(x,y,z)^t$, tiene pixeles con todos los niveles de matiz, $r_H = 6$.

La primera regla se lee de la siguiente manera: si $h_{bc}(i_L,H)$ concentra el valor de los pixeles en el índice de cero y $h_{bc}(i_L,H_2)$ concentra el valor de los pixeles en el índice de cero, entonces los pixeles en $I_b(x,y,z)^t$, son en mayoría anaranjados y amarillos (ver Figura 4.2). Este resultado quiere decir que los pixeles en $I_{bc}(x,y,H)$ y $I_{bc}(x,y,H_2)$ son negros, se debe a que los pixeles en $I_b(x,y,z)^t$ están en el primer cuadrante, teniendo posiblemente matices anaranjados o amarillos. Las demás reglas se leen de la misma manera. La regla dos se refiere a que si los pixeles en $I_{bc}(x,y,H)$ son negros e $I_{bc}(x,y,H_2)$ son blancos, entonces los pixeles en $I_b(x,y,z)^t$ están en el segundo cuadrante. La regla tres se refiere a que si los pixeles en $I_{bc}(x,y,H)$ e $I_{bc}(x,y,H_2)$ son blancos, entonces los pixeles en $I_b(x,y,z)^t$ están en el tercer cuadrante. La regla cuatro se refiere a que si los pixeles en $I_{bc}(x,y,H)$ son blancos e $I_{bc}(x,y,H_2)$ son negros, entonces los pixeles en $I_b(x,y,z)^t$ están en el cuarto cuadrante. La regla cinco se refiere a que si los pixeles en $I_{bc}(x,y,H)$ son blancos e $I_{bc}(x,y,H_2)$ son negros y blancos, entonces los pixeles en $I_b(x,y,z)^t$ están en el segundo hemisferio de H. La regla seis se refiere a que si los pixeles en $I_{bc}(x,y,H)$ son negros e $I_{bc}(x,y,H_2)$ son negros y blancos, entonces los pixeles en $I_b(x,y,z)^t$ están en el primer hemisferio de H. La regla siete se refiere a que si los pixeles en $I_{bc}(x,y,H)$ son negros y blancos, $I_{bc}(x,y,H_2)$ son negros y blancos, entonces los pixeles en $I_b(x,y,z)^t$ tienen todos los niveles en H.

Con base a estas reglas, se pueden definir la cantidad de clases definidas con base a matiz; si la escena tiene niveles de H que cubren un cuadrante, $r_H=2$, porque se pueden tener hasta dos niveles de color en $I_b(x,y,z)^t$, si se tienen dos cuadrantes, $r_H=3$ ya que los colores presentes en $I_b(x,y,z)^t$ tienen tres niveles de matiz y si se tienen todos los cuadrantes, los seis niveles de matiz pueden estar presentes, por tanto $r_H=6$.

Sin embargo, la visión humana define el color a partir de otros componentes aparte de matiz **[**123**,**122**]**. Por lo tanto, se analizó también la información de *S* y *V* para diseñar otro conjunto de reglas que aporten en la determinación de $r_z(t)$. De acuerdo a [123], la componente *S* indica la intensidad de color, y la visión humana es menos sensible a esta característica que *H* o *V*. En otros trabajos relacionados con compresión de colores, esta característica junto con *V* se divide en tres [123] o cinco grupos [121] para realizar una representación de color.

$I_{bc}(x,y,S)^t$ contiene información de saturación dividida en dos partes; la parte de *S* alta donde los matices están bien definidos, y la parte de *S* baja donde el matiz no está muy definido por *H*. No obstante, como se muestra en la Figura 4.2, a pesar de que el matiz no se define bien, aún se preserva cierto nivel de color, incluso se conservan las propiedades de que en un cuadrante de *H* se presentan hasta dos matices, en dos cuadrantes se presentan tres matices y en todos los cuadrantes se mapean los seis matices. Sin embargo, de acuerdo a [123], si $S < 0.2$, se pierde el color (ver Figura 4.4). Dado que la T4CNN no tiene la precisión suficiente para definir si $S < 0.2$ o $S < 0.5$, en este trabajo se plantea que si los valores de *S* en la escena son bajos, entonces se pierde la fidelidad del color a causa de que *H* se encuentra poco definido. Como se observa en la Figura 4.4, aunque es posible tener los seis niveles de matiz cuando *S* es bajo, estos tienden a ser amarillos, cyan y magenta; por lo tanto es difícil que se tengan los seis niveles de matiz si *S* es baja. Adicionalmente, de acuerdo a [123], si los valores de *S* muy bajos ($S < 0.2$), entonces es mejor encontrar la cantidad de regiones por *V* que por *H*, debido a que es posible que la escena se caracterice por tener regiones definidas por niveles de grises y no por color. Un aspecto a considerar en el espacio de color HSV es que al tener niveles de *V* cercanos a cero, los niveles de *S* y *H* no son muy coherentes, ya que en estas condiciones el escenario es obscuro y los colores no están correctamente definidos.

Respecto a *S* y *V*, se puede inferir lo siguiente: si la saturación es baja, los niveles de *H* no llegan a ser los seis niveles ya que los colores pierden fidelidad y el matíz tiende en amarillo, cyan o magenta. Si la saturación es baja y *V* es alto, pueden existir tonos de grises blancos. Si la saturación es baja y V es baja, pueden existir tonos de grises obscuros. Si la escena tiene valores altos y bajos de *S*, entonces la escena contiene tanto colores como escalas de grises. Si la saturación es alta, los colores están bien definidos y no existen tonos

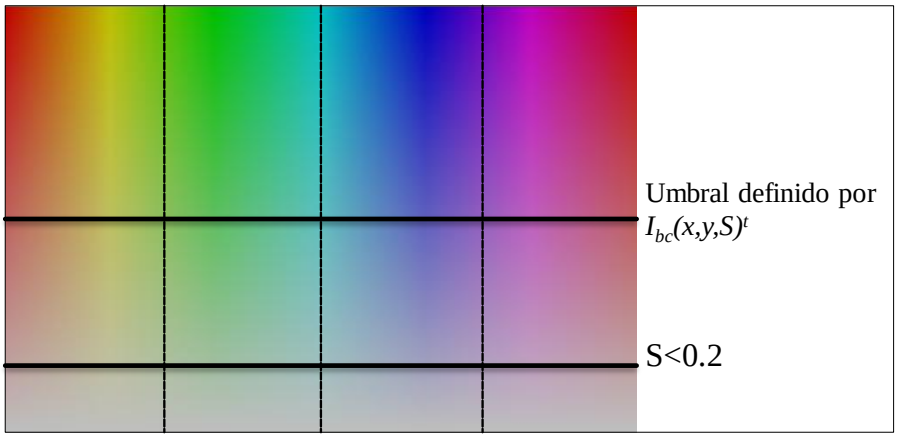


Figura 4.4. Límite definido por saturación

de grises, de forma que la información de V no es relevante. Con base a esto, se definieron las siguientes reglas:

8. Si $h_{bc}(0,S)>0$ & $h_{bc}(1,S)=0$ & $r_H=6$, entonces $I_b(x,y,z)^t$ tiene baja saturación, por lo que los niveles de *H* no están bien definidos, por tanto no se puede confiar que H tenga las seis regiones de matiz, por tanto, con base al análisis realizado, $r_s = -r_H/2$.
9. Si $h_{bc}(0,S)>0$ & $h_{bc}(1,S)=0$ & $h_{bc}(0,V)>0$ & $h_{bc}(1,V)=0$, la saturación es baja y *V* es bajo, por lo tanto existen regiones obscuras que se representan con $r_v = 1$.
10. Si $h_{bc}(0,S)>0$ & $h_{bc}(1,S)=0$ & $h_{bc}(0,V)=0$ & $h_{bc}(1,V)>0$, la saturación es baja y *V* es alta, por lo tanto existen regiones claras que se representan con $r_v = 1$.
11. Si $h_{bc}(0,S)>0$ & $h_{bc}(1,S)=0$ & $h_{bc}(0,V)>0$& $h_{bc}(1,V)>0$, la saturación es baja y *V* puede presentar regiones claras y obscuras que se representan con $r_v = 2$.
12. Si $h_{bc}(0,S)>0$ & $h_{bc}(1,S)>0$ & $h_{bc}(0,V)>0$& $h_{bc}(1,V)>0$, la saturación puede ser alta o baja y *V* puede presentar regiones claras y obscuras, así como diferentes colores de forma que $r_s =0$, $r_v =2$.
13. En cualquier otra condición, $r_s=0$ y $r_v=0$. En esta parte están todas las opciones donde *S* es alto, ($h_{bc}(0,S)=0$& $h_{bc}(1,S)>0$). Dado que en esta condición *H* está bien definido, entonces basta con las regiones definidas por r_H.

Con base a estas reglas, la cantidad de regiones $r_z(t)$ que debe tener el *kmeans* para agrupar las regiones de la imagen de fondo $I_b(x,y,z)^t$ esta dado por:

$$r_z(t) = r_H + r_S + r_V \tag{4.7}$$

Esta cantidad $r_z(t)$ puede cambiar en cada cuadro *t*. De esta manera, se pueden tener de dos a ocho clases basándose en las características de color descritas en el espacio de color HSV. En el siguiente módulo, se van a encontrar los r_z centroides *Cr* en cada cuadro *t*; estos se les denomina $C_r(r_c)^t$, donde $r_c=1,\ldots,r_z(t)$.

4.1.2 Segmentación de objetos estáticos mediante agrupamiento por *kmeans*

Como se mencionó anteriormente, el objetivo principal de la T4CNN es definir la cantidad de clases en que *kmeans* va a segmentar. Una vez que se encuentra $r_z(t)$, KCNN encuentra los $C_r(r_c)^t$ utilizando *kmeans* para posteriormente encontrar las regiones que representen los objetos estáticos con modelos basados en CNN. Sin embargo, el modelo original de *kmeans* obtiene los centros iniciales en forma aleatoria, causando una pérdida de coherencia en la segmentación a lo largo de una secuencia de video ya que en cada cuadro se inicializan los $C_r(r_c)^t$ con valores distintos. Para solucionar este punto, en [133] se propone una modificación en el algoritmos del *kmeans*, donde en el primer cuadro los $C_r(r_c)^t$ se inicializan en forma aleatoria, pero a partir del segundo cuadro los $C_r(r_c)^t$ iniciales son los del cuadro anterior. Esta solución es efectiva si la cantidad de grupos $r_z(t)$ es la misma a lo largo de todo el video. Sin embargo, en este trabajo $r_z(t)$ se determina a partir de la información de color de la escena contenida en la secuencia de video, y la información de color puede estar sujeta a cambios en el fondo, ya que un escenario en una secuencia de video es variante en el tiempo. En consecuencia, $r_z(t)$ puede cambiar su valor si se considera la información de color para determinarlo. Si $r_z(t)$ cambia entre un cuadro y otro, entonces no es posible utilizar los $C_r(r_z)^t$ del cuadro anterior. De esta manera, si $r_z(t)$ puede cambiar a lo largo de la secuencia de video, se propone para KCNN una modificación en la inicialización de $C_r(r_c)^t$ en el *kmeans*, la cual puede ser de tres maneras:

1. Inicialización aleatoria. Si es el primer cuadro de un video o el módulo T4CNN obtiene un $r_z(t)$ que no se había presentado antes, entonces la inicialización de $C_r(r_c)^t$ es aleatoria.

2. Memoria. Si el módulo T4CNN obtiene un $r_z(t)$ distinto a $r_z(t\text{-}1)$, pero ya se había obtenido en un cuadro $t\text{-}t_1$, entonces los $C_r(r_c)^t$ se obtienen a partir de una memoria que se discutirá posteriormente.

3. kmeans anterior. Si el valor de $r_z(t)$ es el mismo que el anterior, entonces los $C_r(r_c)^t$ iniciales son $C_r(r_c)^{t-1}$.

En la Figura 4.5 se puede observar la selección de $C_r(r_c)^t$ iniciales conforme a las tres formas descritas. Una vez que el *kmeans* obtiene los $C_r(r_c)^t$ actuales, se verifica la distancia entre ellos para no tener regiones muy similares o regiones iguales, de la siguiente forma:

$$if \left| C_{ri_c}(r_c)^t \right| - \left| C_{rj_c}(r_c)^t \right| < \varepsilon \quad para \quad i_c \neq j_c \text{, ejecutar de nuevo el } kmeans. \tag{4.8}$$

else, continuar con los $C_r(r_c)^t$ obtenidos.

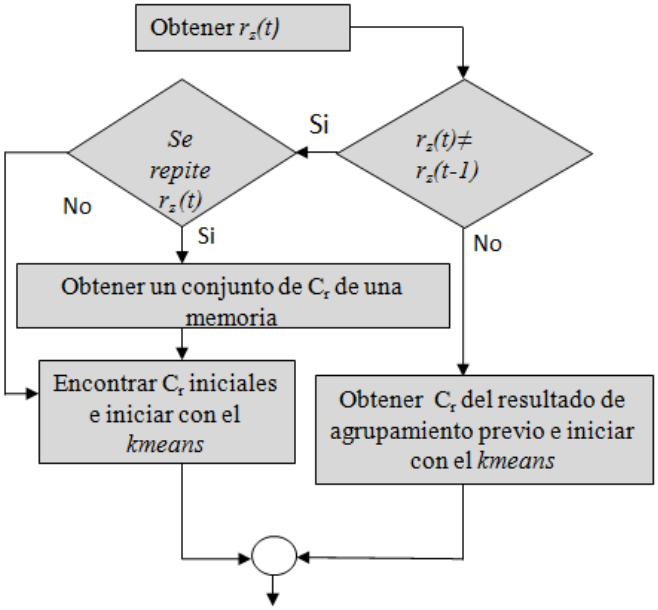


Figura 4.5. Selección de $C_r(r_c)^t$ iniciales para cada *kmeans*.

Para mantener una segmentación coherente en el tiempo y consistente conforme pasan los cuadros, una vez encontrados los centroides $C_r(r_c)^t$, estos son acomodados con base a los niveles de matiz *H*. Esto permite que los $C_r(r_c)^t$ generen regiones con los mismos niveles de grises en todos los cuadros. Un detalle importante que se debe considerar es que tanto en [133] como en KCNN, los centroides iniciales $C_r(r_c)^t$ entran al *kmeans* en coordenadas cartesianas. Por lo tanto, para ordenar los centroides con respecto a la localización de estos en la información de *H*, se realiza lo siguiente:

$$\theta(r_c)^t = \tan^{-1}\left(C_{rx}(r_c)^t / C_{ry}(r_c)^t \right) \quad \theta(r_c)^t = [-\pi/2, \pi/2] \tag{4.9}$$

Basándose en el resultado de $\theta(r_c)^t$ y la localización de $C_{rx}(r_c)^t$ y $C_{ry}(r_c)^t$ en los cuadrantes (*x*,*y*), se calcula la localización de $C_r(r_c)^t$ en *H*. Una vez calculados los niveles de matiz de $C_r(r_c)^t$, estos son ordenados de menor a mayor con base a la información de *H*.

4.2. Módulo de memoria y segmentación

Una vez que se obtienen y se ordenan los centroides, el siguiente paso es el agrupamiento. Con base a [133], el *kmeans* se realiza cada 60 cuadros y en el resto de los cuadros, el agrupamiento se realiza encontrando la distancia euclidiana de cada pixel de $I_b(x,y,z)^t$ con todos los centroides $C_r(r_c)^t$, dado por:

$$E_b(x,y,r_c)=\sqrt{\left(I_b(x,y,X_{hsv})^t-C_{rx}(r_c)^t\right)^2+\left(I_b(x,y,Y_{hsv})^t-C_{ry}(r_c)^t\right)^2+\left(I_b(x,y,Z_{hsv})^t-C_{rz}(r_c)^t\right)^2} \quad (4.10)$$

Para formar los grupos, se busca la distancia euclidiana más corta entre los $r_z(t)$ centroides. Para ello en [133] se propone un método basado en diferentes CNN. La entrada de las TCNN es $E_b(x,y,r_c)$ y la salida $U_s(x,y,r_c)$ van a contener la información de los pixeles que pertenecen a cada clase. En el método propuesto en [**133**], $r_z=4$, por lo tanto $r_c=1,\ldots,4$; y en este método, la segmentación de objetos estáticos está dada por:

$$Is_e(x,y)^t=85\left[U_S(x,y,1)*0+U_S(x,y,2)*1+U_S(x,y,3)*2+U_S(x,y,4)*3\right] \quad (4.11)$$

Hasta la ecuación 4.11 termina el método propuesto en [133]. No obstante, en el caso de KCNN $r_z(t)$ puede cambiar en cualquier cuadro de una secuencia de video; en consecuencia no se puede utilizar la ecuación 4.11 y como se vio anteriormente, es necesario tener una memoria de imágenes y $C_r(r_c)^t$. Esto implica que es necesario generar un módulo que analice el valor de r_z(t) para realizar la segmentación y componer las memorias.

4.2.1 Memoria de centroides e imágenes

Para tener un historial de los resultados del proceso de segmentación, el método propuesto tiene una memoria de centroides y otra de imágenes. Estas memorias se les denominan *LsMem* y *CMem* y se utilizan principalmente para asignar los niveles de grises de las regiones segmentadas y en las condiciones iniciales del *kmeans*. *LsMem* contiene los últimos 10 cuadros segmentados $Is_e(x,y)^t$ y funciona como una memoria *FIFO*. *CMem* contiene los centroides que se utilizaron para segmentar los cuadros $Is_e(x,y)^t$ contenidos en *LsMem* y también funciona como una memoria *FIFO*. Tanto *LsMem* como *CMem* tienen como valores iniciales de ceros. La memoria *CMem* se utiliza para seleccionar los centroides $C_r(r_c)^t$ iniciales cuando el valor de r_z es distinto con respecto al valor anterior. Estos centroides $C_r(r_c)^t$ iniciales son seleccionados a partir de la moda de los $C_r(r_c)^t$

contenidos en *CMem*. La moda de *LsMem* se utiliza para asignar los niveles de grises en el etiquetado de regiones cuando $r_z(t)$ cambia de valor. En la Figura 4.6 se observa una representación de las memorias *LsMem* y *CMem*.

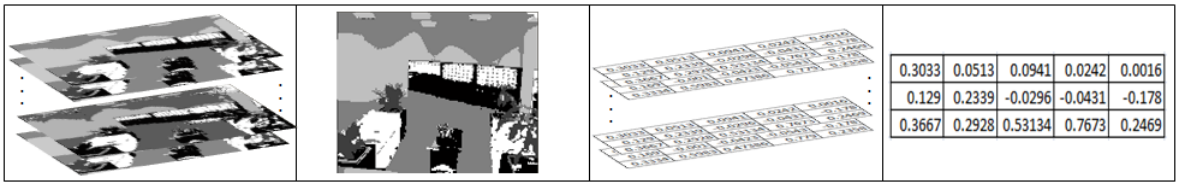


Figura 4.6. Memorias de KCNN. a). *LsMem*. b). Moda de *LsMem*. c). *CMem*. d). Moda de *CMem*.

4.2.2 Segmentación de objetos

En el caso de KCNN, el resultado de la segmentación contenido en $I_{Se}(x,y)^t$ se obtiene etiquetando el resultado de $U_s(x,y,r_c)$ de manera similar a la ecuación 4.11, pero en este caso se asume que $r_z(t)$ puede ser diferente de 4. Para realizar esto, primero se encuentran los niveles de grises con los que se va a etiquetar las regiones. La resolución de los niveles de grises se obtiene a partir de

$$L_s = 256/\left(r_z(t)-1\right) \tag{4.12}$$

La imagen $I_{Se}(x,y)^t$ se obtiene a partir de:

$$Is_e(x,y)^t = L_s\left[U_S(x,y,1)*0+U_S(x,y,2)*1+\cdots+U_S(x,y,r_c)*(r_c-1)+\cdots+U_S(x,y,r_z(t))*(r_z(t)-1)\right] \tag{4.13}$$

El resultado es una imagen que contiene la segmentación representada por diferentes niveles de grises. La ecuación 4.13 solo es válida mientras $r_z(t)$ no cambie. Si $r_z(t)$ cambia, entonces $Is_e(x,y)^t$ se obtiene a partir de:

$$\begin{aligned} Isj\left(x,y,r_S\right) = L_S[&U_s(x,y,1)*\left(0+r_S-R_Z\right)+U_s(x,y,2)*\left(1+r_S-R_Z\right)+\ldots+U_s(x,y,r_c)*(r_c-1+r_S-R_Z)+\ldots \\ &+U_s(x,y,r_z)*(r_z-1+r_S-R_Z)] \end{aligned} \tag{4.14}$$
$$r_S = 1,\ldots,r_z(t)$$

Esta ecuación realiza un ajuste de los niveles de grises cuando cambia el número de regiones $r_z(t)$. $Isj(x,y,r_s)$ es un conjunto de todas las combinaciones posibles de los niveles de grises para el nuevo cuadro segmentado con cantidad de regiones diferentes respecto al cuadro anterior, r_s es un índice del número posible de imágenes con distinta asignación en los niveles de grises, R_Z es un ajuste tal que $r_c+r_s-1<r_z(t)$, entonces $R_Z=0$, si no, $R_Z=r_z(t)$.

De esta forma, $Isj(x,y,r_S)$ se utiliza para designar los nuevos niveles de grises al resultado de segmentación cuando cambia $r_z(t)$. Para realizar esta asignación, se define otro conjunto de imágenes mediante la diferencia entre todas las imágenes en $Isj(x,y,r_S)$ y la moda de *LsMem*, dado por:

$$I_{NS}(x,y,r_S) = \left| Moda(LsMem) - I_{Sj}(x,y,r_S) \right| \tag{4.15}$$

El criterio de selección de la asignación en los niveles de grises es que el valor esperado de las imágenes en $I_{NS}(x,y,j_S)$ indica el grado de similitud entre los resultados de segmentación anterior representados con la moda de *LsMem* y las distintas asignaciones de los niveles de grises en $I_{Sj}(x,y,j_S)$. Este valor esperado es calculado por:

$$\mu_{NS}(r_S) = \frac{1}{K}\sum_{x=1}^{X}\sum_{y=1}^{Y} I_{NS}(x,y,r_S) \tag{4.16}$$

De esta manera, el valor esperado más bajo corresponde a la mayor similitud entre las imágenes de $I_{Sj}(x,y,j_S)$ y la moda de *LsMem*. Por tanto, se estima lo siguiente:

$$m_{js} = \min\left(\mu_{NS}(r_S)\right) \tag{4.17}$$

El resultado en m_{js} indica el índice de la imagen en $I_{Sj}(x,y,j_S)$ que tiene más similitud a la moda de *LsMem* con respecto al criterio del valor esperado. Por tanto, la nueva imagen segmentada estaría dada por:

$$Is_e(x,y)^t = I_{Sj}(x,y,m_{js}) \tag{4.18}$$

En la Figura 4.7a se puede observar la moda de *LsMem* con $r_z=4$, y en las Figuras 4.7b a 4.7d se observan las imágenes de $I_{Sj}(x,y,j_S)$ con los distintos niveles de grises asignados. En este caso, KCNN elige $I_{Sj}(x,y,1)$ ya que el promedio de niveles de grises en esa imagen son más parecidos que en el caso de *LsMem* de la Figura 4.7.

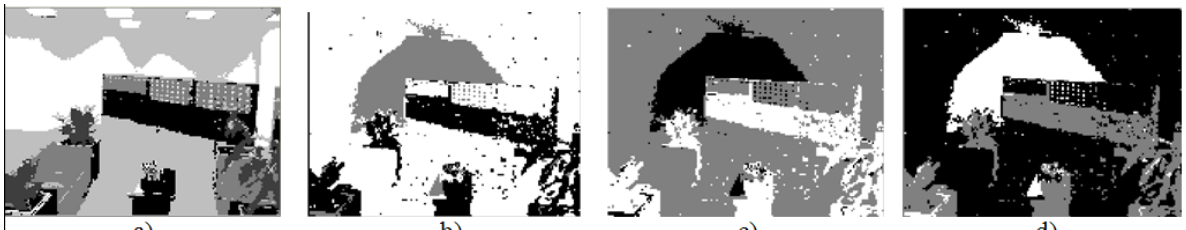


Figura 4.7. Asignación de regiones. a). *LsMem*. b). $I_{Sj}(x,y,1)$. c). $I_{Sj}(x,y,2)$. d). $I_{Sj}(x,y,3)$.

4.3. Resultados cualitativos y conclusiones

Un problema que se presenta en segmentación de objetos estáticos es que no existen en la literatura videos con *ground truth* ni existen bases de datos disponibles que contengan videos que hayan sido diseñados específicamente para segmentación de objetos estáticos. Por lo tanto, las mismas bases de datos utilizadas en el capítulo III serán utilizadas en este y el resto de los capítulos. Como no existen *ground truth* en la literatura para analizar resultados de segmentación de fondo en secuencias de video, resulta complicado obtener métricas confiables respecto al desempeño del método propuesto. Sin embargo, en esta sección se analizarán los resultados cualitativos de algunos ejemplos que describen el funcionamiento de KCNN en videos adquiridos con cámaras fijas y móviles. En el siguiente capítulo se retomará KCNN para ser comparado con un método neuroinspirado de segmentación de objetos estáticos y otros métodos de segmentación de imágenes.

4.3.1 Resultados con videos adquiridos con cámaras fijas

En la Figura 4.8, se pueden observar dos ejemplos de escenarios exteriores y dos de escenarios interiores que se extraen de los videos '*HG*', '*P6*', '*SF*' de *changedetection* y '*P2009_L1_1*' de PETS. Los cuatro videos tienen diversas condiciones comunes que pueden causar problemas en los resultados de segmentación. El video '*HG*' tiene arboles que forman un fondo dinámico además de contener sombras. Los videos '*P6*' y '*SF*' tienen objetos cuyos matices son muy similares en casi toda la escena. Los videos '*SF*' y '*P2009_L1_1*' tienen en una parte de la escena un problema que se le conoce como *cluttering*; es decir, errores de segmentación cuando múltiples objetos en el escenario causan sombra y oclusiones entre sí. A continuación se describirán los resultados de KCNN con estos cuatro videos.

Video 'HG': En este video, $r_z(t)=5$ ya que el método detecta matices azules y verdes por los arboles, amarillos y anaranjados por las banquetas y las líneas pintadas, mientras que la autopista la interpreta con matiz $H \approx 0$ o $H \approx 1$. Con base a esto, KCNN detecta colores en el hemisferio $[0,\pi]$, cumpliéndose la regla 6 por lo tanto $r_H = 3$. KCNN detecta niveles de S bajos, y a causa de las sombras en los arboles y la calle niveles muy variados de V, cumpliéndose la regla 11. Como se puede ver en la Figura 4.8a, los resultados de segmentación son coherentes respecto a los objetos del escenario, ya que son representados

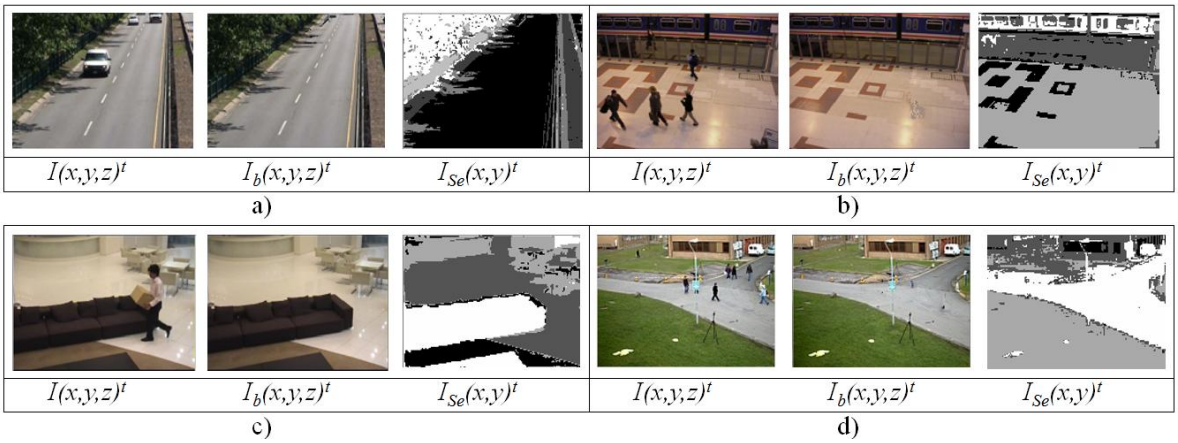


Figura 4.8. Resultados de segmentación de objetos estáticos con los videos de a) '*HG*' *t=120*, donde $r_z(t)=5$. b) '*P6*' *t=110*, donde $r_z(t)=4$. c) '*SF*' *t=120*, donde $r_z(t)=4$. d) '*P2009_L1_1*' *t=120*, donde $r_z(t)=4$.

de forma exitosa los arboles, la calle y las banquetas. Solo existen algunos problemas de sombras.

Video 'P6': En este video, $r_z(t)=4$ ya que el método detecta matices rojos y amarillos. Con base a esto, KCNN detecta colores en el cuadrante $[0,\pi/2]$, cumpliéndose la regla 1 por lo tanto $r_H=2$. KCNN detecta niveles de *S* bajos, y a causa de las paredes obscuras y el piso color claro, los niveles de *V* son muy variados, cumpliéndose la regla 11. Como se puede ver en la Figura 4.8b, los resultados de segmentación son coherentes respecto a los objetos del escenario, ya que son representados de forma exitosa las paredes y el piso. Solo existe ruido en las paredes de reflejos causados por figuras en el piso.

Video 'SF': En este video, $r_z(t)=4$ ya que el método detecta matices rojos y amarillos. Con base a esto, KCNN detecta colores en el cuadrante $[0, \pi/2]$, cumpliéndose la regla 1 por lo tanto $r_H=2$. KCNN detecta niveles de *S* bajos, y a causa del sillón, mesa y tapete obscuros, y el piso claro, V tiene niveles variados, cumpliéndose la regla 11. Como se puede ver en la Figura 4.8c, los resultados de segmentación son coherentes respecto a los objetos del escenario, ya que son representados de forma exitosa las paredes, el piso, el sillón y el tapete. Solo existen errores de segmentación por *cluttering* en las mesas y sillas que se ven alejadas de la cámara.

Video 'P2009_L1_1': En este video $r_z(t)=4$ ya que el método detecta matices verdes y azules. Con base a esto, KCNN detecta colores en el cuadrante $[\pi/2,\pi]$, cumpliéndose la regla 2 por lo tanto $r_H=2$. KCNN detecta niveles de S bajos por la calle y altos por el jardín. Los niveles de V son muy variados, cumpliéndose la regla 12. Como se puede ver en

la Figura 4.8d, los resultados de segmentación son coherentes respecto a los objetos del escenario, ya que son representados de forma exitosa el jardín, la calle y el edificio.

Tanto en los ejemplos de la Figura 4.8 como en otros experimentos se pudo observar que KCNN tiene bastante éxito realizando segmentación de forma coherente de objetos en diferentes tipos de escenarios. Sin embargo, existen algunos problemas relacionados con sombras y reflejos tal como en los ejemplos de '*HG*' y '*P6*'. También existen problemas cuando el escenario tiene *cluttering*.

4.3.2 Resultados con videos adquiridos con cámaras en movimiento

Aún cuando KCNN se diseñó para modelado de fondo adquirido con cámaras estacionarias, este método puede aplicar en videos adquiridos con aplicaciones de cámaras móviles. Como ejemplo, KCNN se puede utilizar en aplicaciones de robótica móvil para analizar los objetos de un escenario y buscar un objetivo. En esta subsección se analizan videos donde un sistema que contiene una cámara móvil se detiene durante un tiempo determinado para analizar la escena. Dado que estas ventanas de tiempo se mantienen fijas durante periodos de tiempo que equivalen a no más de 50 cuadros, para este caso se tomó en consideración que $Lm=5$, para actualizar los centros cada 5 cuadros.

Video Móvil 1: En este video una cámara se mantiene fija en dos ocasiones para reconocer las puertas en un pasillo. En la primer ocasión, la cámara se mantiene estacionaria del cuadro $t=5$ hasta el cuadro $t=35$. Se puede observar en la Figura 4.9 que en el resultado de la segmentación se pudo mantener una segmentación coherente de las puertas contenidas en el pasillo.

Video Móvil 3 y Móvil 6: En estos videos una cámara móvil analiza dos objetos desde varios puntos para identificar los objetos presentes en el escenario, si es que son los objetos de interés o son obstáculos. Para analizarlos, la cámara móvil se mantiene fija durante 25 cuadros en varios puntos. En la Figura 4.10 se pueden ver dos cuadros con diferentes puntos donde los mismos objetos son detectados coherentemente, aunque son etiquetados de forma distinta. En la Figura 4.11, se pueden observar dos cuadros y su resultado de segmentación del video *Móvil 6*. En este video, los objetos fueron segmentados coherentemente, aunque también se muestra que en este video KCNN tuvo mucha sensibilidad a las sombras de la escena.

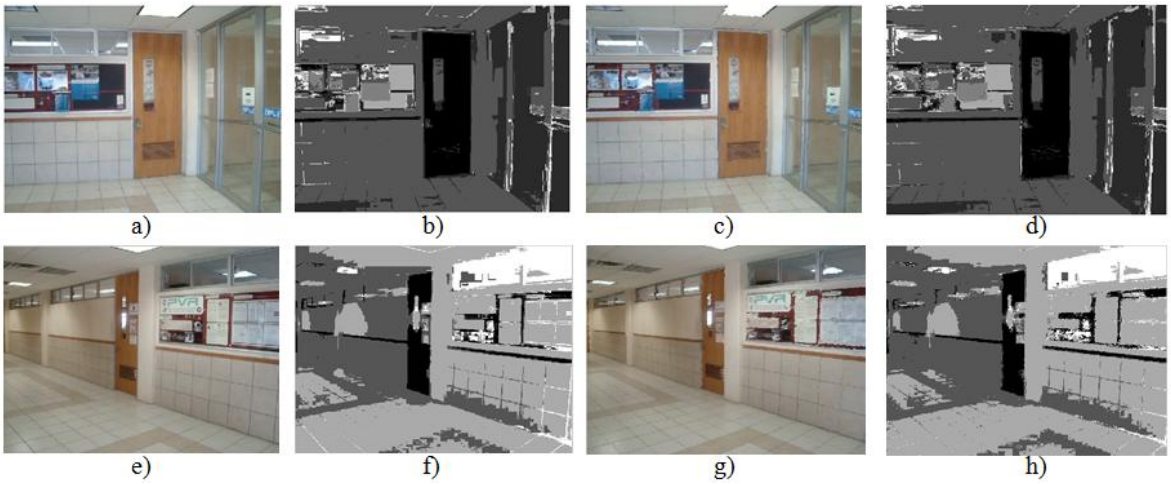


Figura 4.9. Resultados *Video Móvil 1*. a). Fondo del cuadro *t=5*. b). Resultado de la segmentación con KCNN, 4 regiones. c). Fondo del cuadro *t=35*. d). Resultado de la segmentación con KCNN, 4 regiones. e). Fondo del cuadro *t=165*. f). Resultado de la segmentación con KCNN, 4 regiones. g). Fondo del cuadro *t=205*. h). Resultado de la segmentación con KCNN, 4 regiones.

Figura 4.10. Resultados *Video Móvil 3*. a). Fondo del cuadro *t=415*. b). Resultado de la segmentación con KCNN, 4 regiones. c). Fondo del cuadro *t=650*. d). Resultado de la segmentación con KCNN, $r_z(t)=5$.

Figura 4.11. Resultados *Video Móvil 6*. a). Fondo del cuadro *t=992*. b). Resultado de la segmentación con KCNN 4 regiones. c). Fondo del cuadro *t=1002*. d). Resultado de la segmentación con KCNN, $r_z(t)=4$.

4.3.3 Conclusiones

Como se puede observar en los resultados en cámara móvil y cámara fija, KCNN genera una segmentación coherente en diferentes tipos de escenario. Además, en diferentes cuadros de cada video KCNN mantiene la coherencia en seguir los cambios que sufre el escenario. La coherencia en el tiempo de KCNN se analizará en el siguiente capítulo donde se documenta un estudio comparativo entre este método y el que se va a proponer en el siguiente capítulo.

Respecto a los resultados cualitativos, KCNN mostró en este capítulo ser robusto a diferentes tipos de escenarios diferentes tipos de matices y condiciones de iluminación. Sin

embargo, en los diferentes experimentos también mostró tener problemas causados por reflejos y sombras.

V. MÉTODO PERCEPTUAL NEUROINSPIRADO PARA SEGMENTACIÓN DE OBJETOS ESTÁTICOS

Como se vio en el capítulo anterior, la segmentación de objetos estáticos consiste en segmentar un escenario en una secuencia de video en un conjunto de regiones que representen de forma coherente los objetos presentes en el fondo. De acuerdo al capítulo anterior, KCNN puede realizar este análisis, y con base a las pruebas con diferentes videos, se obtuvieron resultados satisfactorios para representar objetos en el escenario. Sin embargo, en las diferentes pruebas se pudo notar que KCNN tiene problemas para detectar objetos en situaciones de cambios de luminancia.

Al tener modelos como CNN, KCNN es un método bioinspirado, pero no un método neuronspirado, ya que no tiene una base neurocomputacional que sustente su forma de procesamiento como un sistema basado en el comportamiento de la corteza cerebral. Los modelos basados en la corteza visual reportados en el capítulo II tienen tiempos de procesamiento muy altos para ser utilizados en secuencias de videos y en ocasiones solo funcionan con escenarios artificiales, por lo tanto, no se pueden utilizar para aplicaciones en visión por computadora. Sin embargo, si se realizan ciertas modificaciones a estos modelos, entonces puede ser posible que se puedan utilizar en escenarios reales y con tiempos de procesamiento más plausibles para análisis de video. Por ejemplo, en el capítulo 2 se presentó el modelo SOMRM, un modelo que simula el proceso de composición de los niveles corticales más bajos de V1, el cual utiliza como entrada un patrón gaussiano y para lograr su objetivo necesita de miles de iteraciones. No obstante, al realizar modificaciones mínimas a SOMRM, entonces es posible que este modelo pueda analizar escenarios reales a velocidades de procesamiento suficientes para aplicaciones en tiempo real.

De acuerdo a la literatura, muchas de las aplicaciones relacionadas al análisis bioinspirado de video se enfoca a la detección de movimiento. Esto contrasta con modelos basados en la corteza visual, ya que si se analiza el estado del arte documentado en el capítulo II, se podrá notar que la mayor parte de los modelos y RNA basados en la corteza visual se enfocan más a estudiar la percepción de objetos. Esta forma de percepción, tiene más cercanía a la segmentación de objetos estáticos que a segmentación de objetos dinámicos. Por lo tanto, una oportunidad de contribución al estado del arte en análisis de

video es proponer un modelo de segmentación de objetos neuroinspirado. Es decir, proponer un método de segmentación de objetos estáticos que se acerque lo más posible a los modelos neurocomputacionales basados en la corteza visual. En consecuencia, en este capítulo se va a proponer un método de segmentación de objetos basado en principios neuroinspirados llamado PBSeg (Segmentación Bioinspirada Perceptual), el cual puede segmentar la información de un cuadro de una secuencia de video en información de color, contornos y luminancia para después integrarlos de manera sistemática y lograr la segmentación de objetos estáticos. Para realizar este proceso de separación e integración de estas características, un conjunto de RNA inspiradas en modelos y redes reportados en el capítulo II se utilizan para simular las capas corticales de la corteza visual.

En la primera sección se profundiza en los modelos neurocomputacionales que sirvieron como inspiración para el desarrollo de PBSeg. En la segunda sección se propone el método PBSeg y cada una de sus capas corticales. En la tercera sección se reporta un estudio comparativo entre PBSeg, KCNN y otros métodos de segmentación. Finalmente en la cuarta sección se reportan las conclusiones.

5.1. Partes de la corteza visual para la detección de objetos y los fenómenos perceptuales asociados

Como se mencionó anteriormente, en la corteza visual existen entre varios, dos procesos de percepción: por un lado la percepción de movimiento y por otro lado la detección de objetos. En la Figura 5.1 se puede observar un esquema de las principales partes de la corteza visual dedicada a la percepción de esta información visual, el cual parte del modelo de Yutzumoto de [32]. Como se mencionó en el capítulo II, para la percepción de objetos y movimiento se involucran las áreas de LGN, V1 y V2. El área V4 se encarga de complementar la percepción y detección de objetos, mientras que MT se encarga de la percepción de movimiento. De acuerdo con diversas teorías y modelos, en los LGN, V1 y V2 se realiza un proceso que incluye la detección y reconstrucción de contornos en mapas de profundidad. Mientras que en V1 y V2 se pasa la información de las superficies para que en V4 se relacione con los contornos y se forme el objeto. Este proceso incluye diversos mecanismos definidos por mapas corticales sensibles a ciertas características de la

información visual, además como se observa en la Figura 5.1, existe una continua retroalimentación entre las áreas V1, V2 y V4.

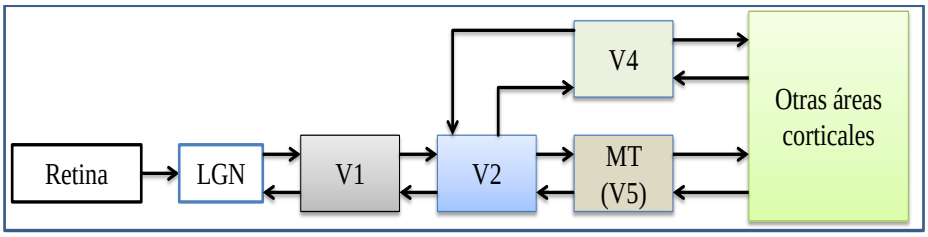


Figura 5.1 Partes de la corteza visual para percepción de objetos y movimiento.

5.1.1 Columnas verticales en V1 y campos receptivos

Hubel y Wisel [31] demostraron con anterioridad que V1 tiene un conjunto de neuronas que son sensibles a ciertas características de los estímulos visuales. Estas neuronas se han conceptualizado como un conjunto de mapas sensibles a orientación, dominancia ocular, color, frecuencia espacial y disparidad. Dichos mapas se describen en la introducción del modelo LISSOM en la subsección 2.4.2. De acuerdo a la anatomía del cerebro, estos mapas neurales se encuentran en un área cortical que se le conoce como columnas verticales. La Figura 5.2, muestra un diagrama de las columnas verticales sensibles a orientación, donde se observa que estas columnas responden directamente de los estímulos de los LGN, los cuales se modelan por la ecuación 2.33. De esta forma, las columnas verticales sensibles a orientación (*OR*) responden a los contornos de los estímulos visuales que se forman en los RF que se pueden modelar como Diferencia de Gaussianos *DoG*.

La parte cortical de las columnas verticales tiene una estructura retinotópica y cada columna está conectada a un elemento de los LGN y a su vez, cada columna tiene una interacción lateral, donde cada columna describe cierta característica de un elemento proyectado en los LGN. Las columnas verticales más estudiadas son las que forman el mapa de orientación (OR), es decir, aquellas que forman un conjunto de células sensibles a la orientación del estimulo de entrada. Existen varios modelos que utilizan estos mapas de OR para simular procesos perceptuales en los niveles más bajos de la corteza visual como la PCNCM [**93**] y PG-LISSOM [140].

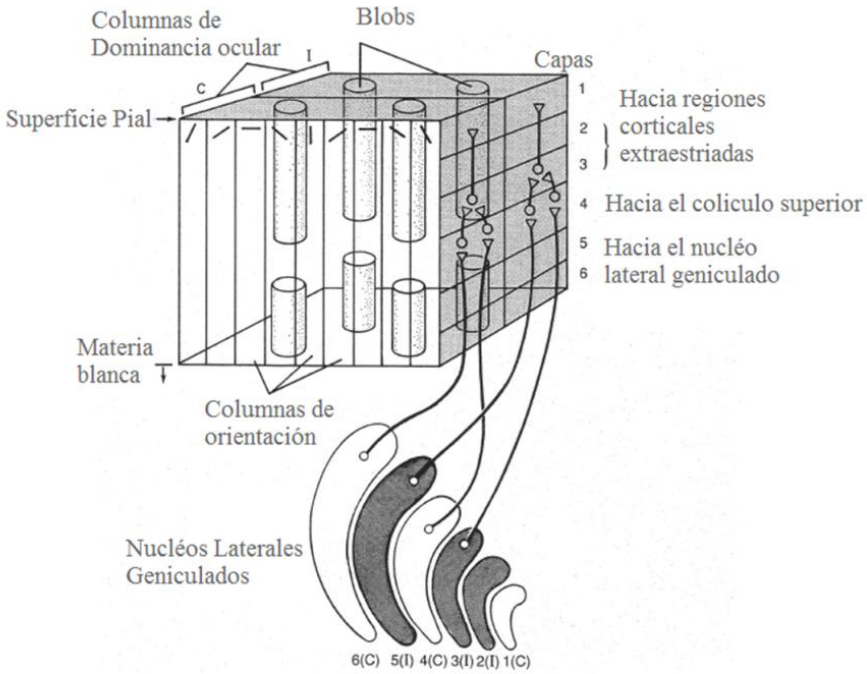


Figura 5.2 Organización columnar en V1. Los LGN tienen aferencias desde la retina, y estos se conectan bajo una estructura reinotópica a las columnas verticales [34].

De acuerdo al modelo pulsante de *Pulse Coupled Network for Countour and Matching* (PCNCM) [93], Cada columna vertical es un elemento de una red que tiene como entrada una serie de impulsos I_θ, H_θ que se disparan en función de la respuesta de los campos receptivos. Ai_θ y Bi_θ forman el potencial de membrana de la columna cuya respuesta es i_θ y representa la orientación de ese punto en el estimulo visual. Estas columnas neurales tienen una interacción lateral como se muestra en la figura 5.3c.

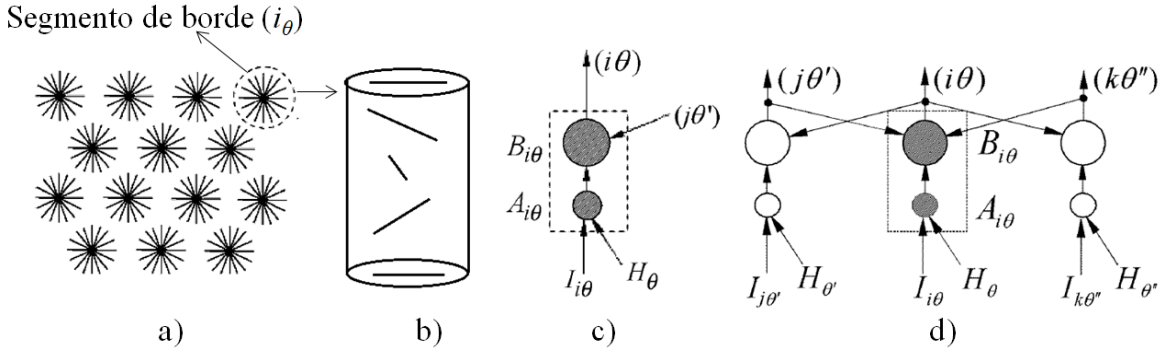


Figura 5.3. Modelo de los mapas OR de la PCNCM [93]. a). Preferencia de orientaciones en una columna. b). Columna neural. c). Estructura de una unidad de la PCNCM. d). Red PCNCM.

Con base al modelo de la PCNCM, se puede notar que las columnas verticales definen un mapa de orientación (OR) tomando como información los estímulos de los LGN que tienen RF dados por *DoG* [93]. El mapa de OR modelado a partir de los impulsos I_θ, H_θ

dados por una red pulsante se utilizan para reconocer los contornos y los patrones de movimiento del estimulo de entrada.

Otro modelo que también utiliza SNN para modelar un mapa de OR es el modelo *Perceptual Grouping LISSOM* (PG-LISSOM) **[**34**,**140**]**, el cual es una red neurocomputacional basada en el modelo de *Eckhorn* y la PCNN, y consta de tres módulos: sinapsis, la agregación y el generador de pulsos. En este modelo, el mapa OR se modela de forma similar al modelo clásico LISSOM y es utilizado para generar el fenómeno de agrupamiento perceptual. En PG-LISSOM, las sinapsis, son un conjunto de integradores que funcionan como una exponencial negativa que va formando los pulsos, lo cual está dado por:

$$s(t) = \sum_{n_g=0}^{t} x(t-n_g)e^{-\lambda n} \tag{5.1}$$

donde *s(t)* es la agregación lineal al tiempo *t*, $x(t - n_g)$ es el pulso de entrada {0,1}, n_g es una ventana de tiempo y λ es la constante de caída. Esta red tiene una activación en V1 dada por:

$$\eta_x(i,j)^{n_L} = \psi\left[\begin{array}{l}\gamma_A \sum_{xy} I(x,y)A(x,y,i,j) - \gamma_E \sum_{i_x j_x} \eta_x(i_x,j_x)^{n_L-1} E_L(i_x,j_x,i,j) \\ \gamma_I \sum_{i_x j_x} \eta_x(i_x,j_x)^{n_L-1} I_L(i_x,j_x,i,j) + \gamma_c \sum_{i_x j_x} \eta_y(i_x,j_x)^{n_L-1} C_L(i_x,j_x,i,j)\end{array}\right] \tag{5.2}$$

Por simplicidad, los pesos aferentes *A(x,y,i,j)* se conectan directamente a la retina, $E_L(i_x,j_x,i,j)$ son los pesos estimuladores, $I(i_x,j_x,i,j)$ son los pesos inhibidores y $C_L(i_x,j_x,i,j)$ son los pesos intracolumnales γ_E y γ_c son constantes de difusión, ψ es la función saturación de la ecuación 2.33. Todos los pesos se actualizan con la regla Hebbiana dada en la ecuación 2.37. Como se muestra en la Figura 5.4, PG-LISSOM representa a V1 en dos capas MAP1 (η_1) y MAP2 (η_2). La capa de MAP1 exhibe un comportamiento de excitación local e inhibición global para cada neurona. Esta capa tiene un esquema auto-organizado para formar un mapa ordenado. La capa MAP2 hace una segmentación con base a una excitación global que genera una integración de contornos y una inhibición local que genera integración de objetos.

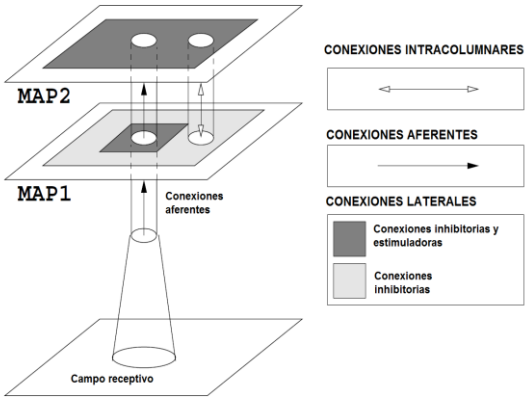


Figura 5.4 Arquitectura de la red PG-LISSOM [140].

En la Figura 5.5 se muestra la *Leaky-Integrator Neuron*, la neurona base de PG-LISSOM en las capas de MAP1 y MAP2, la cual muestra cada una de las sinapsis y la agregación dada por la activación de la ecuación 5.2. El mecanismo de umbral esta dado por:

$$\theta(t) = \theta_{base} + \theta_{abs}(t) + \gamma_{rel}\theta_{rel}(t) \tag{5.3}$$

donde θ_{base} es un umbral base, θ_{abs} es un umbral definido por el periodo pre-disparo γ_{rel} es una constante para el umbral θ_{rel} el cual se refiere al umbral definido en el módulo de retroalimentación inhibidora. Al igual que la PCNN, este módulo se compara con la respuesta de la activación de los mapas corticales para que la neurona dispare o se mantenga, ya que si $\eta(i,j) > \theta(t)$, entonces la neurona dispara. La retina en este modelo es en realidad un estímulo gaussiano para entrenar mapas de orientación similares a los de la Figura 2.41.

De acuerdo al análisis en [140], este modelo resuelve problemas complicados como el triangulo de *kaniza* a partir del modelado de los pulsos en η_1 y η_2 y su influencia en el mapa OR.

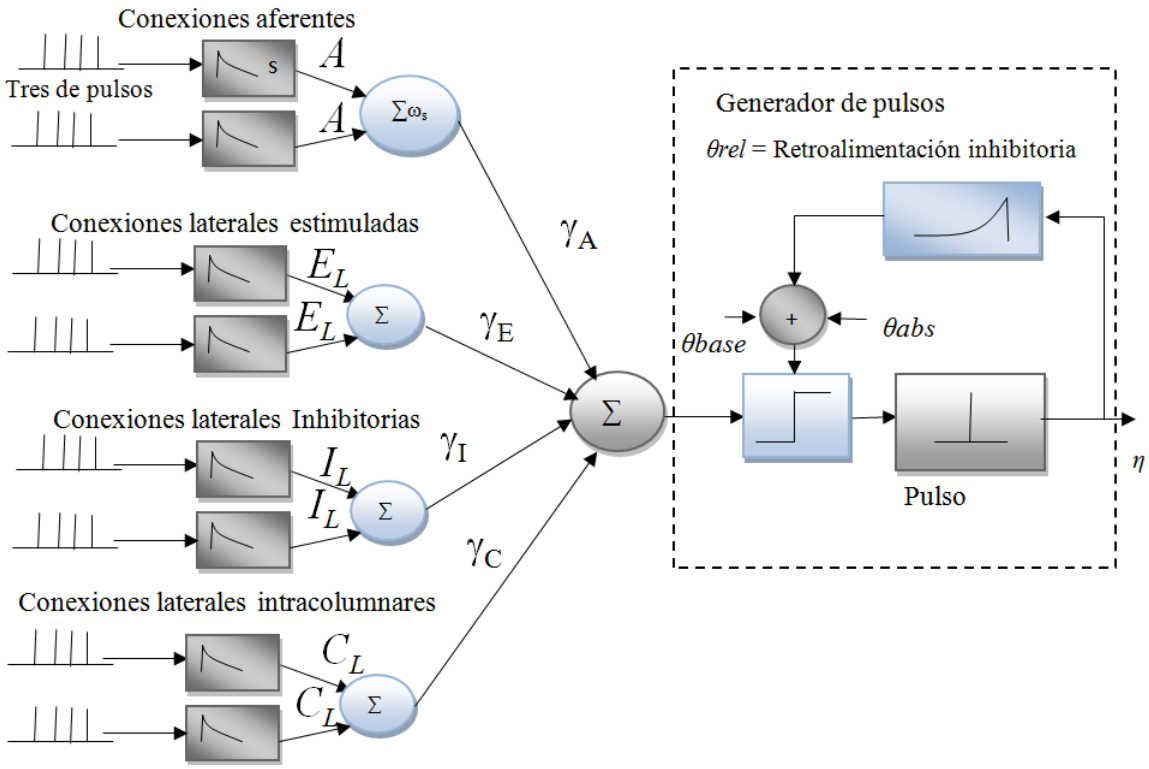


Figura 5.5. *Leaky-Integrator neuron*, modelo utilizado por la PG-LISSOM. El umbral dinámico necesita de un umbral base θ_{base}, una contribución refractoria relativa θ_{rel}, una contribución refractaria absoluta θ_{abs}, ω son los pesos [34].

5.1.2 Simulación de V1

De acuerdo a los modelos descritos anteriormente, diversas características perceptuales como detección de contornos, movimiento y agrupamiento perceptual pueden ser descritos mediante la interacción de una red pulsante y mapas sensibles a orientación. Para probar que la manipulación de un mapa OR con modelos de redes pulsantes puede generar fenómenos perceptuales, se realizó un modelo neurocomputacional en esta tesis que se denominó RF-PCNN, el cual consiste en una simulación de varios niveles que conforman las capas más bajas del procesamiento visual los cuales son:

Capa 1, Retina. Se refiere a la imagen que se utiliza como entrada.

Capa2, RF y LGN. Se definen los RF con la diferencia de gaussiano dadas en la ecuación 2.32 y los LGN con la respuesta dada en la ecuación 2.33. De esta forma, se obtiene una respuesta a los bordes de la imagen proyectada en la retina con las polaridades definidas en los canales ON y OFF. Para esta simulación, $\sigma_c=0.5$, $\sigma s=4.5\sigma_c$.

Capa 3, Columnas de OR. Esta parte simula la parte de V1 totalmente conectada a los LGN y que se encarga de procesar la entrada. Se simula un mapa columnar similar al de la Figura 5.3, donde la respuesta de cada neurona columnar está dada por:

$$V_{or}(i,j) = \exp\left(-\frac{\left[(i_{col} - i_c)\cos(\phi) - (j_{col} - j_c)\cos(\phi)\right]^2}{\sigma_a^2} - \frac{\left[(i_{col} - i_c)\cos(\phi) + (j_{col} - j_c)\cos(\phi)\right]^2}{\sigma_b^2} \right) \quad (5.4)$$

donde σ_a es la amplitud a lo largo de la columna, σ_b es la amplitud a lo ancho, i_{col}, j_{col} los índices de la neurona columnar, i_c, j_c el centro de la columna. Se puede seleccionar cualquier dimensión para el tamaño de la columna, pero en el caso de RF-PCNN, $i_{col}=1,..,31$, $j_{col}=1,...,31$, $\sigma_a=15$, $\sigma_a=1$, en cada columna, $\phi=0, \pi/4, \pi/2, 3\pi/4$. En la Figura 5.6 se observa la respuesta de una columna basandose en la ecuación 5.4, la cual como se muestra en la Figura 5.3b va combinando cada barra de orientación.

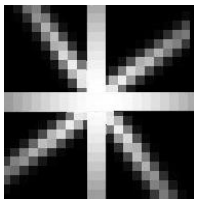


Figura 5.6. Columna vertical del modelo descrito.

Capa 4, Red pulsante. Se realiza una codificación pulsante de la respuesta de las columnas verticales para simular V1. Para esto, se utiliza la PCNN (ecuaciones 2.19 a 2.22), la cual crea una serie de pulsos que se van acumulando con la ecuación 2.23.

En la Figura 5.7 se muestra la arquitectura de RF-PCNN que se utilizará para analizar la parte de columnas verticales y su relación con fenómenos derivados de la percepción visual. Las imágenes utilizadas fueron la ilusión de *Hermman* y el mensaje '*bad eyes'*, las cuales se muestran en la Figura 5.8. La primera consiste en una ilusión que como se puede apreciar en la Figura 5.8a, se forman en la intersecciones de las líneas blancas la ilusión de círculos grises que aparecen y desaparecen. En la Figura 5.8b se observa un patrón visual que aparentemente no contiene ninguna información; no obstante, si se entrecierran los ojos, se puede apreciar la frase de '*bad eyes'*.

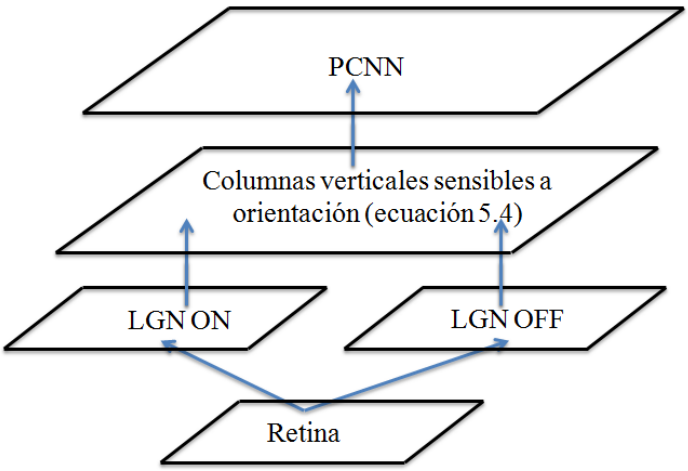
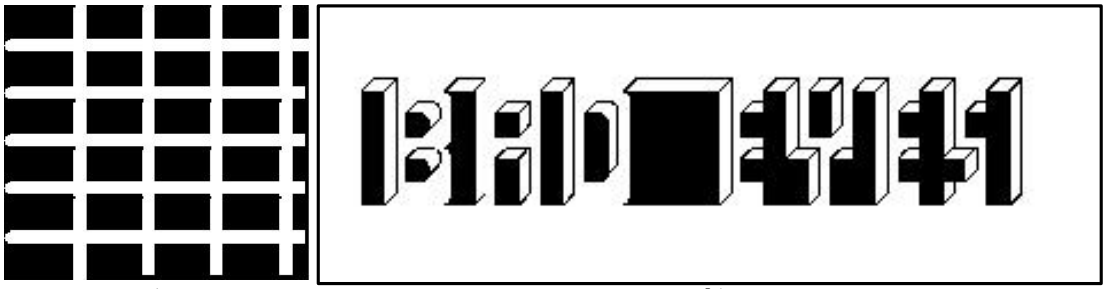


Figura 5.7. Arquitectura del modelo propuesto RF-PCNN.



a) b)

Figura 5.8. Ilusiones. a). Reja de *Herman*. b). Ilusión de los ojos chinos.

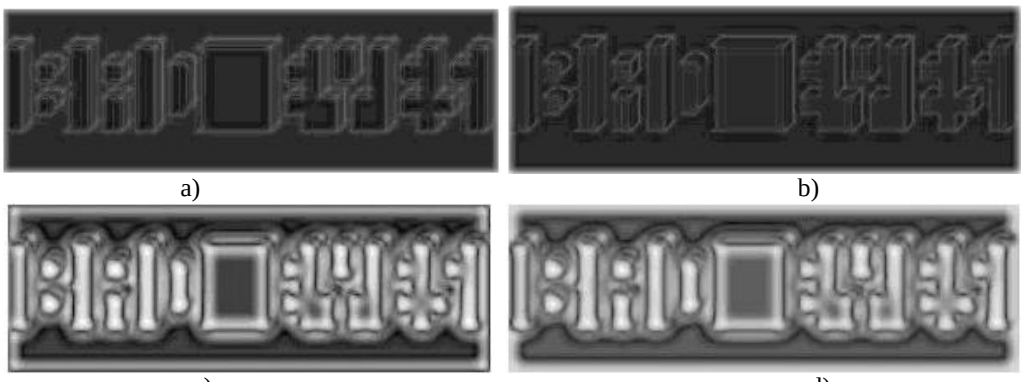
De acuerdo a [141], la hipótesis respecto a los círculos grises que el lector puede apreciar visualmente en la Figura 5.8a es que se originan debido a un efecto secundario no deseado causado desde los RF, ya que el canal ON crea cuatro dosis de potencial de acción inhibitorio por la parte de alrededor en las intersecciones de las barras blancas, mientras que en los límites de dichas barras solo hay dos inhibiciones. El resultado puede ser que se debilite en la actividad del potencial en el canal ON. Para comprobar esto, se utilizó la imagen de la Figura 5.8a como entrada al modelo RF-PCNN con 200 iteraciones. En la Figura 5.9, se puede observar la salida en ambas canales después de 200 iteraciones, donde se muestra que el canal ON crea dichos círculos en las intersecciones de las barras. Tal como se aprecia en la Figura 5.9, estos círculos se muestran de forma clara en el canal ON, mientras que el canal OFF muestra un patrón un poco distinto. Tal como se plantea en [142], las inhibiciones del canal alrededor son muy marcadas en los bordes de las barras, lo cual se aprecia en ambos canales. No obstante, en las intersecciones estas inhibiciones se suman formando un efecto como se ve en las flechas verdes, donde en el canal ON tienen

fuerte efecto inhibitorio, aislando la intersección y creando el circulo. En el canal OFF la polaridad se activa en forma inversa, por lo tanto no aísla la intersección.

a) b)

Figura 5.9 Resultado de procesar RF-PCNN con la imagen de la ilusión de *Hermann* de la Figura 5.8a. a). canal ON. b). canal OFF.

La ilusión de los ojos chinos se percibe al entrecerrar los ojos y observar la frase '*bad eyes*', de otro modo no se puede observar nada. Esto quiere decir que en el momento de entrecerrar los ojos, los RF que parten de las RGC hasta los LGN tienen un mapeo diferente de la información de los conos y bastones. Al utilizarse esta imagen como entrada del modelo RF-PCNN, se obtiene como salida en cada canal (ON, OFF) imágenes similares a la entrada como se muestra en la Figura 5.10a y 5.10b. Sin embargo, si se modifica el parámetro de σ_c a 4.5 e RF-PCNN para simular el efecto de borrosidad que se da cuando se hacen los 'ojos chinos', se crea un patrón muy distinto al de la Figura 5.10a y 5.10b, lo cual se puede apreciar en la Figura 5.10c y 5.10d. Esta nueva salida que produce la RF-PCNN, crea un patrón visual que cualquier lector lo relaciona con la frase '*bad eyes*'.



a) b)

c) d)

Figura 5.10. Salida de la RF-PCNN utilizando como entrada la imagen de los ojos chinos. a).b). Canales ON y OFF con $\sigma_c=0.5$, $\sigma_s=4.5\sigma_c$. c).d). Canales ON y OFF con $\sigma_c=4.5$, $\sigma_s=4.5\sigma_c$.

Con base a estos experimentos, se puede ver que el modelo RF-PCNN hace procesamiento similar al que sucede al nivel de los RF y las columnas verticales sensibles a orientación. Este modelo muestra que redes como PCNN son suficientes para sustentar

algunos principios de percepción si se utiliza bajo un esquema basado en el funcionamiento de la corteza visual. En ambos casos, los RF formaron los bordes y los RF sensibles a OR estimularon dichos bordes cuando ϕ coincide con el ángulo de estos. No obstante, las esquinas pueden ser estimuladas en distintos valores de ϕ, como se observa en la Figura 5.11, donde la salida de las columnas verticales en $\phi=\pi/4$ y las esquinas forman un estímulo que también se presenta en otros valores de ϕ. Este estímulo es se activa con las esquinas pero inhibitorio en la intersección y al ser codificado por la PCNN en series de tiempo da como salida los círculos grises de la Figura 5.9.

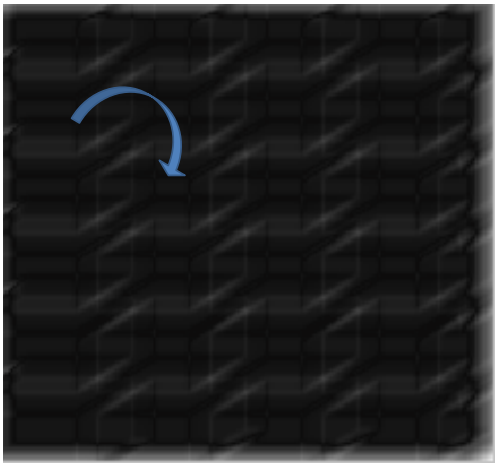


Figura 5.11. Columna vertical donde $\phi=\pi/4$.

Con base a RF-PCNN y diversas teorías de percepción derivada de modelos neurocomputacionales, en la sección 5.2 se va a plantear un nuevo modelo de segmentación de objetos en secuencias de video.

5.1.3 LISSOM Tricromático (TLISSOM)

Como se mencionó en la sección 2.4.2, LISSOM es un conjunto de modelos basados en auto-organización de pesos fundamentado en los niveles más bajos de la corteza visual. Además, previamente se mencionó que los modelos LISSOM generan como resultado diversos mapas de características como el de orientación (OR), dominancia ocular (OD), frecuencia espacial (SF), y color (CR). LISSOM Tricromático (TLISSOM) es un modelo que tiene como objetivo construir el mapa de color (CR) mediante la auto-organización de los pesos y activación de V1 ante un estímulo de entrada. Para ello, el modelo LISSOM toma la misma arquitectura definida en la sección 2.4.2 y la combina bajo una estructura

que se modela a partir de las teorías de campos receptivos y células simples que se forman en el tálamo, V1 y V2 [143].

En la Figura 5.12 se puede ver la arquitectura general del modelo, el cual consiste de 8 unidades ON/OFF, seis canales en la retina y dos capas corticales que definen la actividad de V1 y V2. La retina contiene la imagen de entrada en el espacio de color LMS (*Long-Medium-Short*), el cual simula la respuesta de los conos en la retina de forma que los conos L son sensibles al rojo, M al verde y S al azul. Al igual que en muchos modelos relacionados con percepción de color a nivel de la corteza visual, este tiene 4 tipos de RF ON/OFF, los cuales son:

RF L/M ON y OFF: Estos son sensibles a longitudes de onda Rojo y Verde.
RF M/L ON y OFF: Estos son sensibles a longitudes de onda Verde y Rojo.
RF B/(M+L) ON y OFF: Estos son sensibles a longitudes de onda Amarillo y Azul.
Luminancia: Sensibles a la intensidad de luz proyectada en la retina.

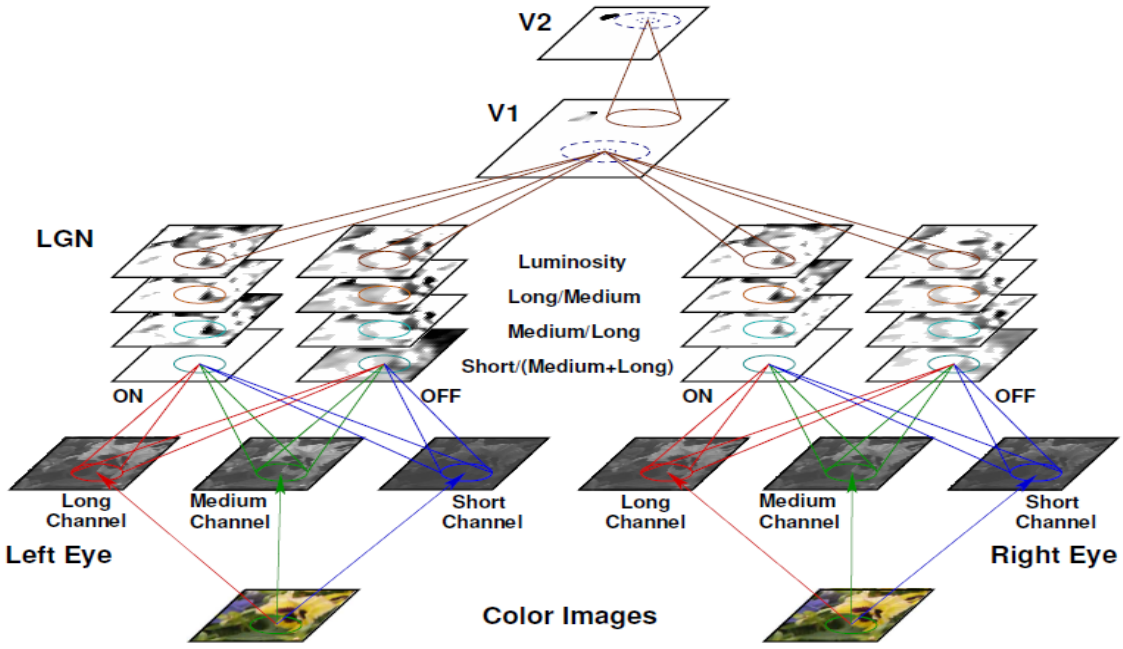


Figura 5.12 Modelo LISSOM Tricromático (TLISSOM) **[**143**]**.

La información de los RF pasa a V1 el cual tiene el mismo modelado que LISSOM dado por la activación cortical definida en las ecuaciones 2.36 y 2.37. Sin embargo, en el caso de este modelo, cada tipo de RF tiene su propia activación en V1. Tanto en V1 como en V2 se modela un mapa de OR y otro de CR. Como se mencionó en el capítulo II, los

mapas en V1 y V2 son conjuntos de neuronas sensibles a ciertas características de la escena. De acuerdo a **[**143**,**144**,**145**]**, los primeros pasos en los procesos de percepción inician con la interacción de la información de los mapas como CR, OR, OD, DY. Con base a esto, se puede concluir que los fenómenos perceptuales modelan no solo mediante columnas verticales sensibles a OR como en la subsección anterior, también a través de la relación de otras características como color, frecuencia espacial y profundidad.

Una forma de combinar el esquema de TLISSOM y las redes pulsantes PG-LISSOM, RF-PCNN o incluso LEGION, es diseñar una red que modele los RF sensibles a luminancia y color, luego utilizar una capa auto-organizada que represente las células simples de V1 y después que un conjunto de redes pulsantes procesen la información de estas redes auto-organizadas para obtener la información de orientación y color. La interacción de estas características debe ser utilizada para retroalimentar las capas mencionadas para lograr un mejor resultado.

Si se desarrolla un método de segmentación con todos los elementos mencionados, entonces se podrá obtener un esquema de segmentación que contenga diferentes partes y conceptos de los modelos reportados en el capítulo II, pero aplicado a solucionar problemas relacionados a visión por computadora y análisis de video. En la siguiente sección se abordará dicho método de segmentación el cual fue diseñado para segmentar los objetos estáticos o de fondo.

5.2. Segmentación de objetos estáticos basado en modelos de la corteza visual

Respecto a los modelos descritos de RF-PCNN, PG-LISSOM, TLISSOM, así como el modelo LAMINART descrito en el capítulo II, se puede concluir lo siguiente:

- Como se mencionó en la sección 5.1, los mecanismos perceptuales de RF-PCNN se obtuvieron a partir de la descomposición del resultado de los RF en sus distintas orientaciones y codificación de pulsos.
- De acuerdo a PG-LISSOM y RF-PCNN, la codificación de pulsos puede modelar mapas de orientación en V1 que permiten simular fenómenos perceptuales.

- De acuerdo al modelo TLISSOM, los niveles más bajos de la corteza visual contribuyen en los procesos de percepción mediante la correlación entre los diferentes mapas de color de OR, CR y otros como DO y gradiente [143].
- El modelo TLISSOM plantea que los RF que mapean la información que viene de los conos y bastones se mapea en los *RF L/M, M/L, S/(L+M)* y luminancia. La información de estos RF es la entrada a V1 y V2 los cuales modelan la información con base a la auto-organización de pesos [143].
- El modelo LAMINART sustenta diversos procesos de percepción bajo un esquema donde V1 realiza una detección de contornos, mientras que V2 y V4 se retroalimentan entre sí para combinar la información de las superficies con los contornos para formar la percepción de objetos [41**]**.
- En diversos modelos, se plantea que las capas más bajas de la corteza visual tienen un comportamiento que se puede asemejar a los modelos auto-organizados.

Con base a estos puntos, se desarrolló un modelo para segmentación de objetos estáticos bioinspirado. En la Figura 5.13 se puede observar un diagrama del método propuesto el cual en esta tesis se denomina PBSeg (*Perceptual Bioinspired Segmentation*), donde la entrada es un cuadro de una secuencia de video en RGB. La información de cada cuadro se descompone en luminancia y color en el módulo de los LGN. En este módulo, existe un parámetro que indica la sensibilidad para filtrar la información de color y luminancia. La información de color se descompone en los LGN formando cuatro canales sensibles a los niveles de rojos, amarillos, verdes y azules, mientras que la información de luminancia se descompone en los canales ON/OFF de los RF para generar los contornos de los objetos presentes en el escenario. De esta manera, los LGN forman 6 canales; cuatro de color y dos de contornos. Esta información pasa al módulo V1, el cual funciona como una capa auto-organizada basada en RESOM que aprende los objetos de fondo para estabilizar la información de color y contornos. La salida de RESOM modela el fondo del escenario y es mapeada en un conjunto de campos receptivos de orientación (RFO) que mejoran el filtrado de color y estimulan los contornos definidos por los canales ON/OFF de luminancia. La salida de los RFO de los canales de luminancia pasa a otra etapa de V1 donde una red neuronal pulsante genera una detección de bordes que se van estimulando si

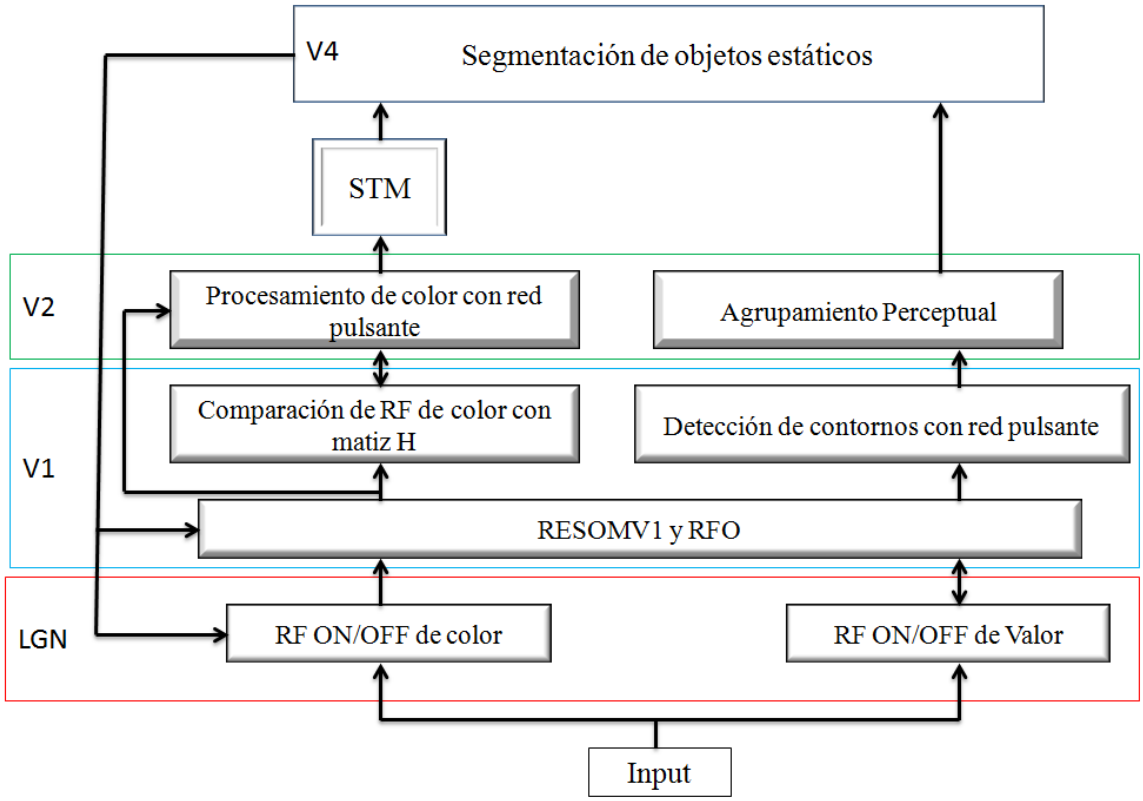


Figura 5.13. Esquema del método PBSeg.

estos se repiten a lo largo de la secuencia de video o se inhiben si solo se presentan en pocos cuadros. Con esta respuesta, se puede generar un agrupamiento perceptual que permite integrar los contornos de los objetos de fondo e inhibir bordes de objetos dinámicos y ruido.

La información de los RFO relacionados a los canales de color pasa a V2 para que una red neuronal pulse tres veces para descomponer la información de color en distintas regiones binarizadas. La red neuronal pulsante ajusta ciertos parámetros basándose en un análisis de matiz en V1. La información de color que genera la red neuronal pulsante se almacena en una memoria de corto plazo (STM) que guarda los últimos diez resultados de la red pulsante de V2. En el módulo de V4, el primer paso es obtener un promediado de la información guardada en la STM y el resultado se binariza con una NTCNN para realizar una primer propuesta de segmentación. Esta propuesta es comparada con el resultado del agrupamiento perceptual para verificar si existe coherencia entre la información de color y de luminancia. Si en la comparación existen poca coherencia, V4 reajusta parámetros en los

LGN y RESOM hasta que encuentre consistencia entre color y luminancia. En las siguientes subsecciones se explicarán cada uno de estos módulos.

5.2.1 Entrada

En diversos modelos meurocomputacionales, la entrada representa la retina. En TLISSOM, la retina realiza una transformación del espacio de color RGB al espacio de color LMS. Aunque en esta tesis se experimentó con este espacio de color, no se consideró en el diseño del método PBSeg, ya que es necesario calcular parámetros de los dispositivos de adquisición del video o imagen [143]. Existen otros modelos que mapean de RGB a LMS que parten del modelo XYZ, como en [146]. No obstante, en este caso tampoco resulta útil la conversión dada en [146] ya que se reduce el contraste en la información de rojos y verdes, generando mayor dificultad en la detección de regiones. Por lo tanto, en PBSeg la entrada es el cuadro $I(x,y,z)^t$ en el espacio de color RGB.

5.2.2 Núcleos Laterales Geniculados (LGN)

Como se mencionó anteriormente, los RGC proyectan la información de la retina a los LGN y estos a su vez a la corteza visual y otras áreas del cerebro. Los RGC se unen mediante un conjunto de células llamadas campos receptivos (RF) que se activan o desactivan en función de la estimulación proveniente de la retina. Dichos RF se modelan con un conjunto de gaussianas con diferentes desviaciones estándar y dirección. Los LGN están localizados en una parte del cerebro que se le conoce como tálamo y tiene tres tipos de células, las células tipo Magno, Parvo y Konio **[**147**,**72]. En TLISSOM se toman los tres tipos de células LGN, ya que los conos y bastones se conectan de la siguiente forma:

- Los conos Sh (sensibles a los matices azules) se conectan a las células konio.
- Los conos L y M se conectan a las células parvo.
- Los bastones sensibles a los cambios bruscos de movimiento se conectan a las células magno.

En la zona parvocelular, se tienen RF C/A, que se pueden modelar a partir de *DoG* que están dadas por la ecuación 2.32. Existen dos tipos de canales, los RF centro L alrededor M (L/M) y los RF centro M y alrededor L (M/L). Mientras que la zona koniocelular tiene otro tipo de RF llamado coextensivo [146], el cual se basa en dos gaussianas con la misma varianza pero con signo distinto [143]. En esta zona se estimula

con las longitudes de onda azules (Sh) y se inhibe con las longitudes de onda amarillas (L+M). El caso de la zona magnocelular, se encuentran los RF centro/alrededor que son sensibles a luminancia. Como se mencionó al inicio del capítulo II, los LGN no solo envían las señales a la corteza visual, también reciben fibras de otras áreas para retroalimentar procesos de atención.

Debido a que el modelo PBSeg no utiliza el espacio LMS, los RF L/M y M/L serán implementados con información de los canales rojo y verde. De esta forma, los RF quedan clasificados como R/G, G/R y B/(R+G). Como se aprecia en la Figura 5.14, los RF de PBSeg se modelan como centro-alrededor (C/A), de tal manera que todos los RF se van a dar por *DoG*. Sin embargo, estos RF están dados por la convolución de la *DoG* con una imagen a escala de grises de un canal, lo cual es un problema en los RF de color ya que requieren al menos dos canales de color como se aprecia en la Figura 5.14. Esto se debe a que en la literatura es muy extensivo el modelado de RF con imágenes a escalas de grises o para analizar problemas relacionados con luminancia, pero es muy poca la información referente a RF de color, los cuales tienen como entrada dos canales.

Sin embargo, en esta forma de modelado los *DoG* representan la respuesta neuronal de los RF como un sistema lineal invariante en el tiempo (LTI), entonces es posible modelar la diferencia de dos convoluciones, donde cada una es una combinación lineal entre un canal de color y una gaussiana. Para plantear los RF de color, se va a considerar lo siguiente; como se mencionó anteriormente, las *DoG* de los RF están dados por ecuaciones como la ecuación 2.32 y la respuesta de un RF es la convolución del estímulo de entrada con el RF como se aprecia en la ecuación 2.33. Por lo tanto, si se considera una gaussiana dada por:

$$Ga_r = \frac{\exp\left(-\frac{(x-x_c)^2+(y-y_c)^2}{\sigma_r^2}\right)}{\sum_{uv}\exp\left(-\frac{(u-x_c)^2+(v-y_c)^2}{\sigma_r^2}\right)}, \qquad r=c,s \tag{5.5}$$

donde σ_r es la amplitud de la gaussiana Ga_r, r se refiere a la gaussiana de centro (r=c) o alrededor (r=s), u,v es el índice de la respuesta del campo receptivo, del receptor (x,y) y

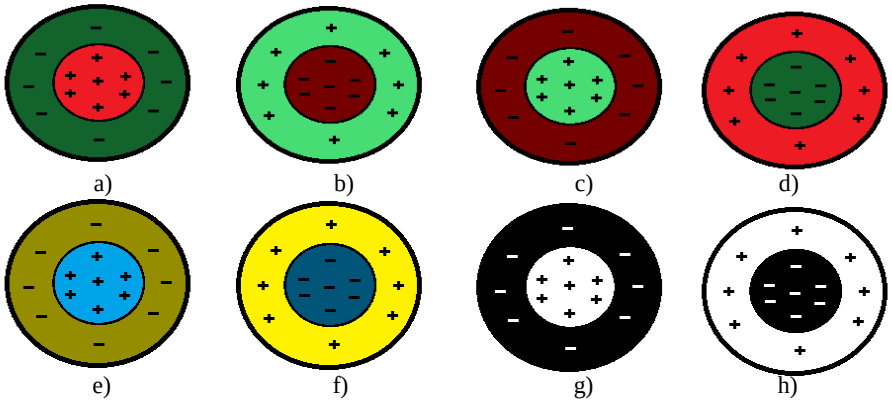


Figura 5.14. Campos receptivos RGC-LGN de BPSeg. a). R/G ON parvocélular. b). R/G OFF parvocélular. c). G/R ON koniocélular. d). G/R OFF koniocélular. e). B/(R+G) ON. f). B/(R+G) OFF. g). Luminancia ON. h). Luminancia OFF.

centro (x_c,y_c). La respuesta de los RF dada por la ecuación 2.32 se puede replantear como $L = Ga_c - Ga_s$, donde $\sigma_c=0.5$, y $\sigma_s=4\sigma_c$.Con base a la ecuación 2.33, la respuesta de los RF y la respuesta de los LGN están dados por:

$$RF(a,b) = I(x,y) * \left[Ga_c(x,y,a,b) - Ga_s(x,y,a,b) \right] \tag{5.6}$$

$$\varsigma(a,b) = \psi\left(RF(a,b) \right) \tag{5.7}$$

(a,b) es la posición del campo receptivo. Se puede observar que la ecuación 5.6 y la combinación lineal $\psi(.)$ de la ecuación 2.33, tienen como entrada una sola imagen *I(x,y)*. Por tanto, para aplicar esta respuesta en los RF de color, se utilizaron algunas propiedades de linealidad para asumir lo siguiente:

$$RF(a,b) = I(x,y) * Ga_c(x,y,a,b) - I(x,y) * Ga_s(x,y,a,b) \tag{5.8}$$

$$\varsigma(a,b) = \psi\left(RF(a,b) \right) \tag{5.9}$$

En la Figura 5.15, se observa que la respuesta de los LGN en los canales ON/OFF utilizando la ecuación 5.6 y la ecuación 5.8 es la misma. Este mismo principio se va aplicar a los RF color, ya que en los RF de color interactúan dos canales, ya que de acuerdo a **[**72**,**144**,**148**]**, los RF se distribuyen de la siguiente manera:

RF R/G o L/M: La longitud de onda roja es sensible a la gaussiana del centro Ga_c, mientras que la longitud de onda verde es sensible a la gaussiana del alrededor Ga_s.

RF G/R o M/L: La longitud de onda verde es sensible a la gaussiana del centro Ga_c, mientras que la longitud de onda roja es sensible a la gaussiana del alrededor Ga_s.

RF B/(R+G) o Sh/(L+M): La longitud de onda azul es sensible a la gaussiana del centro Ga_c, mientras que la longitud de onda amarilla es sensible a la gaussiana del alrededor Ga_s.

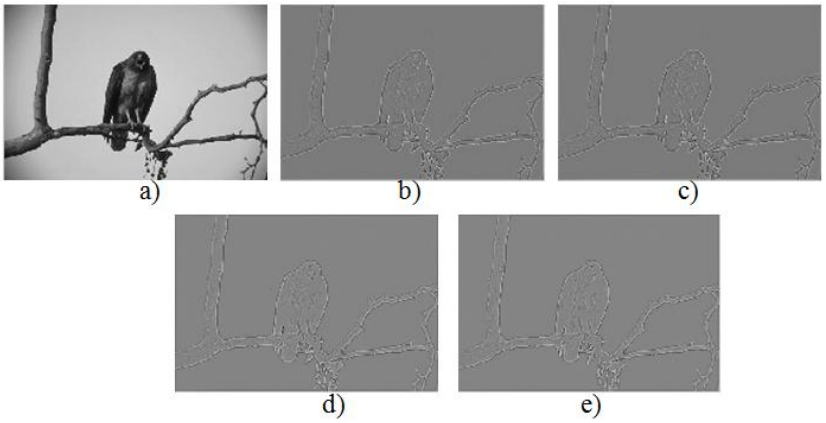


Figura 5.15. Ejemplo del manejo de la respuesta lineal de los RF. (a). Imagen *Test Image* #42049 (Ave). (b). LGN ON con ecuación 5.6. (c). LGN ON con ecuación 5.8. (d). LGN OFF con ecuación 5.6. (e). LGN OFF con ecuación 5.8.

Basándose en estos puntos, y la ecuaciones 5.6 y 5.8, los RF de color en PBSeg quedan modelados como:

$$RG(a,b) = I(x,y,R) * Ga_c(x,y,a,b) - I(x,y,G) * Ga_s(x,y,a,b) \quad (5.10)$$

$$\varsigma_{rg}(a,b) = \psi\left(RG(a,b)\right) \quad (5.11)$$

$$GR(a,b) = I(x,y,G) * Ga_c(x,y,a,b) - I(x,y,R) * Ga_s(x,y,a,b) \quad (5.12)$$

$$\varsigma_{gr}(a,b) = \psi\left(GR(a,b)\right) \quad (5.13)$$

$$BY(a,b) = I(x,y,B) * Ga_c(x,y,a,b) - \left[I(x,y,G) + I(x,y,R)\right] * Ga_s(x,y,a,b) \quad (5.14)$$

$$\varsigma_{by}(a,b) = \psi\left(BY(a,b)\right) \quad (5.15)$$

donde *RG(a,b)* son los RF ON/OFF R/G, *GR(a,b)* son los RF ON/OFF G/R y *BY(a,b)* son los RF ON/OFF B/(R+G). En el caso de los RF ON, las gaussianas Ga_c *y* Ga_s tienen signo positivo, mientras que en el caso de los RF OFF, las gaussianas Ga_c *y* Ga_s tienen signo negativo. En el caso de los RF de luminancia, estos están dados por:

$$Lum(a,b) = I(x,y,V) * \left[Ga_c(x,y,a,b) - Ga_s(x,y,a,b)\right] \quad (5.16)$$

$$\varsigma(a,b) = \psi\left(Lum(a,b)\right) \quad (5.17)$$

donde *Lum(a,b)* es el RF ON/OFF del canal de luminancia que se representa por la información de valor del actual cuadro y $\varsigma(a,b)$ la respuesta de los LGN. En la Figura 5.16 se puede observar un ejemplo de la implementación de los LGN de color con la imagen *Training Image 232038* de la base de datos de Berkeley, donde los LGN muestran cierta

sensibilidad a algunas longitudes de onda; los LGN RG ON y GR OFF muestran más sensibilidad a los niveles de rojos, los LGN GR ON y RG OFF muestran más sensibilidad a los niveles de verde, el LGN BY ON muestra más sensibilidad al azul y el RF BY OFF muestra más sensibilidad al amarillo, mientras que los RF de valor son sensibles a los contornos.

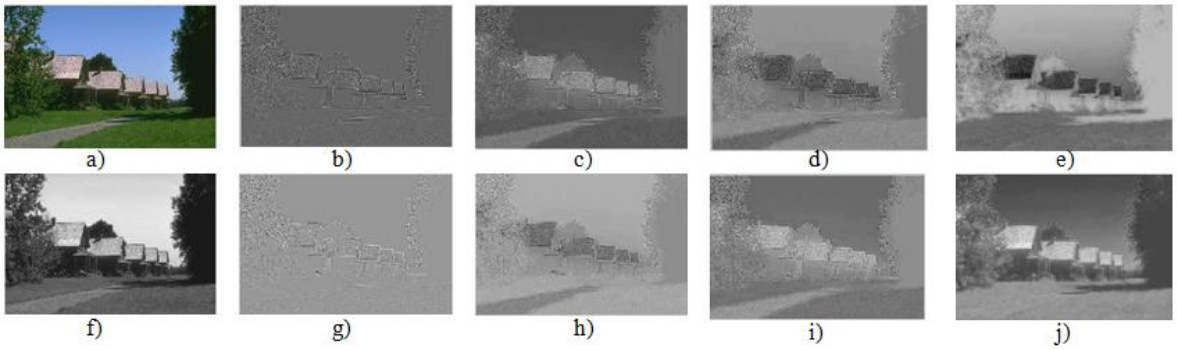


Figura 5.16. Respuesta de los LGN de color y luminancia en la imagen *Training Image 232038* (vereda). a).Imagen original. b). RF Lum ON. c). LGN RG ON. d). LGN GR ON. e). LGN BY ON. f). Componente de valor de la imagen original. g). LGN Lum OFF. h). LGN RG OFF. i). LGN GR OFF. j). LGN BY OFF.

Para analizar la sensibilidad al color de los LGN RG, GR y BY, se realizaron una serie de experimentos con imágenes que van cambiando en matiz y saturación, pero con V constante. En la Figura 5.17 se observa la respuesta de los LGN de color con dos imágenes (Figuras 5.17a y 5.17e) cada una con V constante pero H y S distintos. En ambos casos, la respuesta de los RF de color es la misma ya que dichos RF son invariantes a los niveles de luminancia y V. Este comportamiento se debe a que el resultado de los RF puede dar cualquiera de las siguientes opciones:

1. Si en un pixel $I(x_1,y_1,z)^t$ los valores de los tres canales de color son iguales, entonces, de acuerdo a las ecuaciones 5.10 a 5.15 en los LGN el resultado para todos los LGN es cero.
2. Si en un pixel $I(x_1,y_1,z)^t$ el valor de los tres canales es distinto, entonces, para todos los LGN los resultados serán diferentes de cero.
3. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal rojo es mayor que el valor del cana verde, entonces $RG(a_1,b_1)/ON>0$, $GR(a_1,b_1)/ON<0$, $RG(a_1,b_1)/OFF<0$, $GR(a_1,b_1)/OFF>0$.

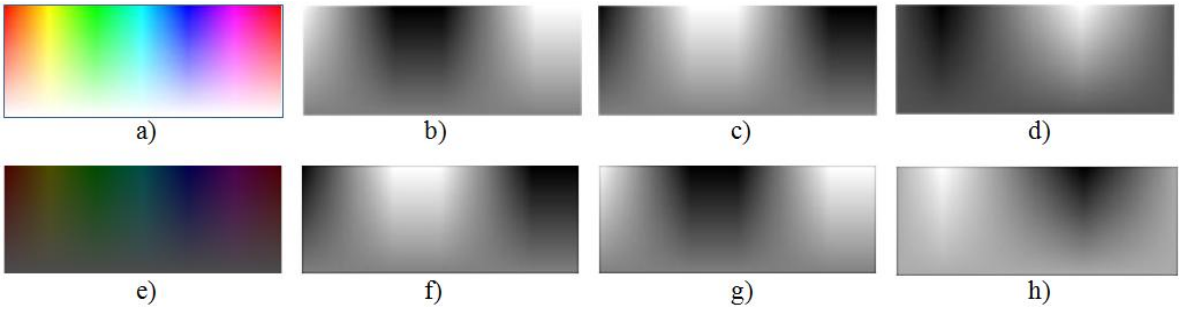


Figura 5.17. Respuesta de los LGN de color en imágenes con matiz y saturación variables. En los LGN θ_l=0 y θ_u=*1* a). Imagen con V=1. b). LGN RG ON. c) LGN GR ON. d). LGN BY ON. e). Imagen con V=0.3. f). LGN RG OFF. g). LGN GR OFF. h). LGN BY OFF.

4. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal verde es mayor que el valor del cana rojo, entonces $RG(a_1,b_1)/ON<0$, $GR(a_1,b_1)/ON>0$, $RG(a_1,b_1)/OFF>0$, $GR(a_1,b_1)/OFF<0$.
5. Si en un pixel $I(x_1,y_1,z^{)t}$ el valor del canal azul es mayor que la suma de los canales verde y azul, entonces $BY(x_1,y_1)/ON>0$ y $BY(x_1,y_1)/OFF<0$.
6. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal azul es menor que la suma de los canales verde y azul, entonces $BY(x_1,y_1)/ON<0$ y $BY(x_1,y_1)/OFF>0$.

Para los puntos 2 y 3, los resultados entre $RG(a_1,b_1)/ON$, $GR(a_1,b_1)/OFF$ y entre $GR(a_1,b_1)/ON$, $RG(a_1,b_1)/OFF$ son iguales si σ_c y σ_s son los mismos para los RF $RG(a_1,b_1)$ y $GR(a_1,b_1)$.

De esta forma, si un pixel $I(x_1,y_1,z)^t$ tiene los mismos valores en las tres componentes R,G y B, los resultados en todos los LGN de color será cero, de otro modo el resultado será distinto de cero. Aunado a esto, conforme el color en un pixel $I(x_1,y_1,z)^t$ se acerque a R, G o B, entonces el valor absoluto del resultado en los LGN se acercará más a 1. En consecuencia, los LFN serán variantes a S. Respecto a información de matiz, los LGN funcionan de la siguiente manera: los LGN RG ON y GR OFF tendrán valores mayores a cero en pixeles con niveles de H rojo y menores a cero en matices verdes. Los LGN RG OFF y GR ON tendrán valores mayores a cero en pixeles con niveles de H verdes y menores a cero en matices rojos. Los LGN BY ON tendrán valores mayores a cero en pixeles con niveles de H azules y menores a cero en matices amarillos. Los LGN BY OFF tendrán valores mayores a cero en pixeles con niveles de H amarillos y menores a cero en matices azules. Los LGN son invariantes a V, ya que esta componente se asocia con luminancia y brillo; esta información se elimina con las diferencias de las convoluciones de las ecuaciones 5.10 a 5.15, ya que la luminancia y brillo en un pixel $I(x_1,y_1,z)^t$ es la misma

en los tres canales de color [143]. La función $\psi(.)$, es la función no lineal de la ecuación 2.33 que normaliza la respuesta de los LGN con los límites inferior θ_l y superior θ_u. Con la ecuación 2.33, los LGN generan una respuesta que se puede observar en las Figuras 5.15 y 5.17, donde los pixeles con bajo nivel de *S* dan a 0.5, mientras que en los niveles de color, si el RF da un valor negativo, entonces $0<\varsigma<0.5$, y si los RF dan valores positivos, $0.5>\varsigma>1$. Esta transformación de los LGN es útil no solo para visualizar el resultado de los RF, también se pueden manejar los parámetros de θ_l y θ_u tener la sensibilidad deseada en los RF para detectar color. Por ejemplo, en la Figura 5.18 los LGN tienen umbrales que fueron seleccionados en θ_l=0.5 y $\theta_u=1$, de forma que los valores negativos y ceros de los RF tienen respuesta de cero en los LGN, dejando que los LGN actúen como filtros de color.

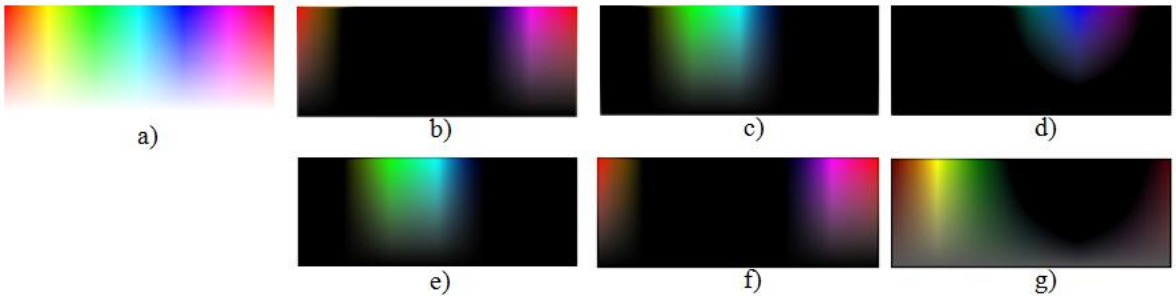


Figura 5.18. Respuesta de los LGN de color, donde θ_l=0.5 y $\theta_u=1$. a). Imagen original. b). LGN RG ON. c). LGN GR ON. d). LGN BY ON. e). LGN RG OFF. f). LGN GR OFF. g). LGN BY OFF.

Mejoramiento de los LGN como filtros de color. De acuerdo a los puntos analizados, se puede concluir que los LGN de color tienen una sensibilidad que se asemeja al comportamiento de un conjunto de filtros de color. No obstante, se puede observar en las Figuras 5.17 y 5.18 que los LGN no tienen los mismos intervalos de funcionamiento en los cuales dejan pasar el color (esto es notorio en los LGN BY ON y BY OFF). Esto sucede porque las convoluciones $I(x,y,z)*Ga_r(x,y,a,b)$ generan resultados con rangos distintos en $r=s$ y $r=c$ debido a que las gaussianas Ga_c y Ga_s tienen amplitudes distintas, perdiendo proporcionalidad en la información del color en cada RF cuando se obtienen las convoluciones en las ecuaciones 5.10 a 5.15. Por lo tanto, para mejorar el desempeño de los LGN como un módulo de análisis de color, se realiza una normalización del resultado de la convolución de las gausianas Ga_c y Ga_s de la siguiente forma:

$$sRF(a,b) = I(x,y,z)^t * Ga_r(x,y,a,b) \tag{5.18}$$

$$SRF(a,b) = sRF(a,b) - \min(sRF(a,b)) \tag{5.19}$$

$$RFs(I(x,y,z)^t, Ga_r) = SRF(a,b) / \max(SRF(a,b)) \tag{5.20}$$

Esto permite escalar el resultado de las convoluciones para que los rangos de $I(x,y,z)*Ga_r(x,y,a,b)$ sean los mismos en $r=s$ y $r=c$, y en consecuencia el filtrado de los LGN tenga el mismo rango de funcionamiento en todos los canales. De esta manera, la respuesta de los RF mejorados (RF_m) está dado por:

$$RGON = RFs(I(x,y,R)^t, Ga_c) - RFs(I(x,y,G)^t, Ga_s) \quad (5.21)$$

$$RGOF = RFs(I(x,y,R)^t, -Ga_c) - RFs(I(x,y,G)^t, -Ga_s) \quad (5.22)$$

$$GRON = RFs(I(x,y,G)^t, Ga_c) - RFs(I(x,y,R)^t, Ga_s) \quad (5.23)$$

$$GROF = RFs(I(x,y,G)^t, -Ga_c) - RFs(I(x,y,R)^t, -Ga_s) \quad (5.24)$$

$$BYON = RFs(I(x,y,B)^t, Ga_c) - RFs(I(x,y,G)^t + I(x,y,R)^t, Ga_s) \quad (5.25)$$

$$BYOF = RFs(I(x,y,B)^t, -Ga_c) - RFs(I(x,y,G)^t + I(x,y,R)^t, -Ga_s) \quad (5.26)$$

Las ecuaciones 5.21 a 5.26 tienen las mismas reglas en cuanto al resultado de los valores y su interpretación, es de la siguiente forma:

1. Si en un pixel $I(x_1,y_1,z)^t$ los valores de los tres canales de color son iguales, entonces, de acuerdo a las ecuaciones 5.10 a 5.15 en los RF el resultado para todos los RF es cero.
2. Si en un pixel $I(x_1,y_1,z)^t$ el valor de los tres canales es distinto, entonces, para todos los RF los resultados serán diferentes de cero, pero en los intervalos de [-0.5,0) y (-0,0.5).
3. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal rojo es mayor que el valor del canal verde, entonces $RGON>0$, $GRON<0$, $RGOF<0$, $GROF>0$.
4. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal verde es mayor que el valor del canal rojo, entonces $RGON<0$, $GRON>0$, $RGOF>0$, $GROF<0$.
5. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal azul es mayor que la suma de los canales verde y azul, entonces $BYON>0$ y $BYOF<0$.
6. Si en un pixel $I(x_1,y_1,z)^t$ el valor del canal azul es menor que la suma de los canales verde y azul, entonces $BYON<0$ y $BYOF>0$.

Para los puntos 2 y 3, los resultados entre *RGON*, *GROF* y entre *GRON*, *RGOF* son los mismos si σ_c y σ_s son iguales. Para PBSeg, los LGN se diseñaron para obtener el mismo parámetro de sensibilidad θ_l y que la salida esté en el intervalo de [0,1]. Por tanto, la respuesta de los LGN está dada por:

$$RF_{\Theta} = RF_{\lambda} - \Theta \tag{5.27}$$

$$\varsigma_{\lambda}(RF_{\lambda}) = \tfrac{1}{2}\left(\left|RF_{\Theta}\right| - \left|RF_{\Theta} - 1\right| + 1\right) \tag{5.28}$$

donde RF_{λ} es un campo receptivo dado por cualquiera de las ecuaciones 5.21 a 5.26, λ es un índice de campo receptivo ($\lambda=R,G,B,Y$), Θ tiene valores de [-0.5,0.5] e indica la sensibilidad en el filtrado del color de manera similar a θ_l, la ecuación 5.28 es una función que normaliza el resultado en el rango de [0,1]. La respuesta de los LGN en PBSeg funciona de manera similar a la ecuación 2.33, y tiene mayor plausibilidad con las teorías modernas de percepción ya que tal como se mencionó en el capítulo II, los LGN filtran información no necesaria para mejorar el proceso de atención visual; en el caso de PBSeg, los LGN se diseñaron como filtros de color que descartan con el parámetro Θ información redundante. Al igual que los RF de color, los canales de los LGN de color son variantes a H y S pero invariantes a V. En la Figura 5.19 se puede observar la respuesta del módulo LGN con distintos valores de Θ; cada LGN ς_{λ} tiene diferente sensibilidad a los distintos niveles de color. Si Θ=-0.5, los ς_{λ} son sensibles a grises y filtran muy pocos colores, de manera que para ciertos colores al menos tres LGN se estimulan, por ejemplo, el azul se repite en $\varsigma_{R,}\varsigma_{G}$ y ς_{B}. En caso de que Θ=0, los ς_{λ} filtran bandas de colores más definidas, de manera que para ciertos colores al menos dos LGN se estimulan, por ejemplo, el azul se repite en ς_{G} y ς_{B}. Finalmente, si Θ=0.5, los ς_{λ} filtran niveles de grises y colores con saturación baja, y cada ς_{λ} es sensible a diferentes tipos de colores, por ejemplo ς_{B} solo es sensible al azul.

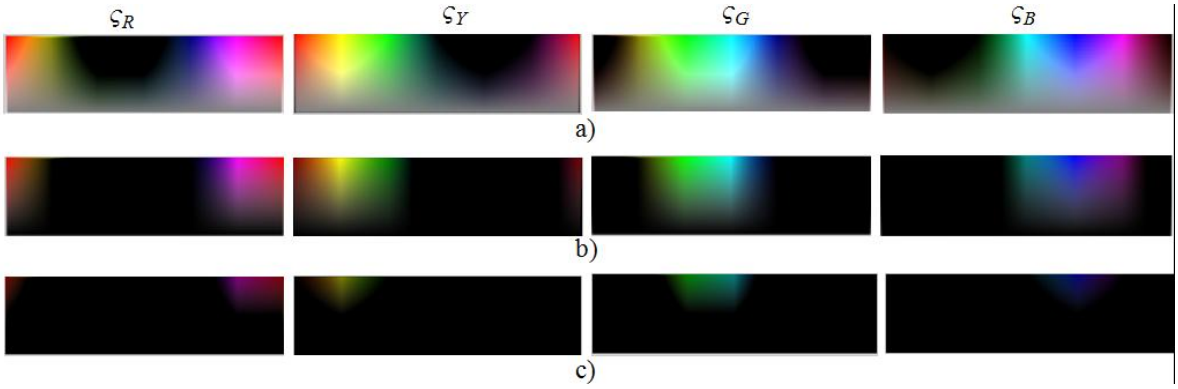


Figura 5.19. Respuesta de los LGN codificado en color con las ecuaciones 5.21 a 5.26. a). Respuesta con Θ=-0.5. b). Respuesta con Θ=0. c). Respuesta con Θ=0.5.

Para el método de PBSeg (Figura 5.13), los módulos LGN se definen con las ecuaciones 5.21 a 5.26, con valores iniciales para $\varsigma_{\lambda}(RF_{\lambda})$ como Θ=0, para que actúen como

filtros sensibles a rojos, amarillos, verdes y azules tal como se ve en la Figura 5.19. En la Figura 5.20 se pueden observar dos ejemplos de la respuesta de los LGN con Θ=0, donde se puede apreciar que los canales de color son selectivos a diferentes tonos de color. Por ejemplo, en el cuadro de '*P2001_1*', los edificios estimulan los LGN rojo y amarillo, el jardín estimula los LGN verde y amarillo, la calle el LGN azul y el cielo, al ser blanco no estimula ningún LGN. En el caso del video '*MO*' las sillas rojas estimulan fuertemente el LGN rojo y un poco el LGN amarillo, la pared estimula los LGN amarillo y un poco el rojo. Las sillas color cyan estimulan los LGN verde y azul.

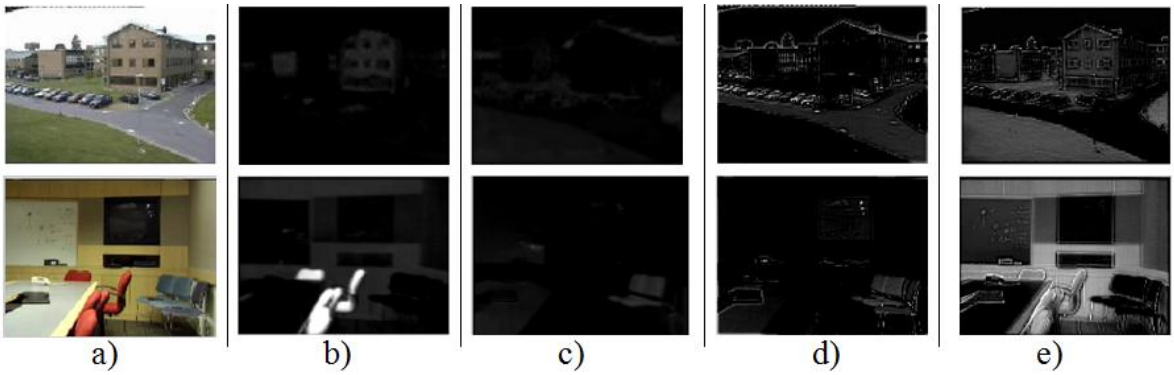


Figura 5.20. Resultados de procesar los LGN en cuatro canales de color. a) Cuadro t=10 de '*P2001_1*' y '*MO*'. b). LGN RG ON. c). LGN GR ON. d). LGN BY ON. e). LGN BY OFF.

En las siguientes subsecciones se abordarán otros módulos de PBSeg y también como a partir del resultado en módulos superiores se mueve el valor de Θ para mejorar la coherencia en los resultados de la segmentación.

5.2.3 Primer capa del módulo V1

El funcionamiento del módulo V1 se basa en la teoría desarrollada en los modelos LISSOM y LAMINART, por lo tanto, V1 tiene una estructura auto-organizada que detecta contornos mediante pulsos e inicia un análisis de color. Este módulo tiene dos capas como se puede ver en la Figura 5.13. La primera es una red cortical basada en RESOM que representa las interacciones topográficas de los modelos LISSOM, pero en PBSeg se utilizará para aprender el fondo en la secuencia de video. En esta capa, también se realiza un análisis de varios filtros de orientación (RFO). En la segunda capa de V1, la información de los LGN de luminancia es procesada por una red neuronal pulsante para detectar los contornos. Adicionalmente, en esta segunda capa de V1 se inicia un análisis de color.

Estas capas asemejan su comportamiento a PG-LISSOM, TLISSOM y LAMINART. El objetivo en PG-LISSOM es modelar un mapa de OR para obtener un agrupamiento perceptual. Para TSLISSOM el objetivo de V1 es modelar los mapas de OR, CR y DO. Para LAMINART, el objetivo en V1 es encontrar los contornos de los objetos y separarlos de la información de intensidad luminosa de superficies. Aparentemente los modelos mencionados no tienen nada en común, sin embargo, si se analizan los antecedentes del capítulo II de LAMINART y los antecedentes de PG-LISSOM y TLISSOM se puede concluir lo siguiente:

1. La entrada de V1 tiene un conjunto de células simples que tienen una estructutra auto-organizada y están correlacionadas con los mapas neurales sensibles a distintas características como orientación y color.
2. Como sugiere PG-LISSOM y LAMINART, en V1 se van formando los contornos de los objetos para formar diversos fenómenos como el agrupamiento perceptual.
3. Como sugieren LAMINART y TLISSOM, existe en V1 un preprocesamiento de la información de contenido de los objetos. El modelo LAMINART recurre solo a analizar intensidades de tonos de grises ya que no considera color y en TLISSOM se forma el mapa de CR.

Con base a estos tres puntos se diseñó el modulo de V1, el cual se describe a continuación.

Capa RESOM para entrada en módulo V1. De acuerdo a diversos modelos neuroinspirados, la primera parte del modulo V1 es una etapa cortical que simula las interacciones topográficas en V1 similar a lo que se describe en **[**34**,**143]. La entrada a esta primer etapa de V1 es la respuesta de los RF de luminancia y color [145]. Sin embargo, para PBSeg en esta primera etapa cortical en V1 se implementa con una red RESOM que se denomina RESOMV1 ya que es el mismo modelo del capítulo III pero utilizado ahora para modelar la equivalencia de las células simples 4 y 3B[21] (Figura 2.6) en PBSeg. Desde el punto de vista del procesamiento, RESOMV1 es una primera etapa cortical que mapea la información de color y luminancia de los LGN de manera que se pueda modelar el fondo de

[21]En los modelos LISSOM no se mencionan estas capas corticales, pero si son mencionadas en LAMINART como conjunto de células simples.

de los canales RF de color (canales $\varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B$) y luminancia (ON,OFF). Otro objetivo con RESOM es reducir el ruido que suele presentarse entre cuadros consecutivos.

La entrada a V1 son la respuesta de los LGN que se transforman de *(a,b)* a *k* y se agrupan en un arreglo $Ic(k,z_c)$, donde $z_c=\{\varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B,ON,OFF\}$, $\varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B$ los canales de color y *ON*, *OFF* son los canales de luminancia. Como se observa en la Figura 5.21, RESOMV1 tiene una red de cuatro neuronas para cada canal de los LGN; en consecuencia se tienen seis redes RESOM. La cantidad de neuronas por canal se estableció en cuatro al igual que en el capítulo III ya que es la cantidad suficiente para modelar correctamente el fondo pero evitando altos costos computacionales.

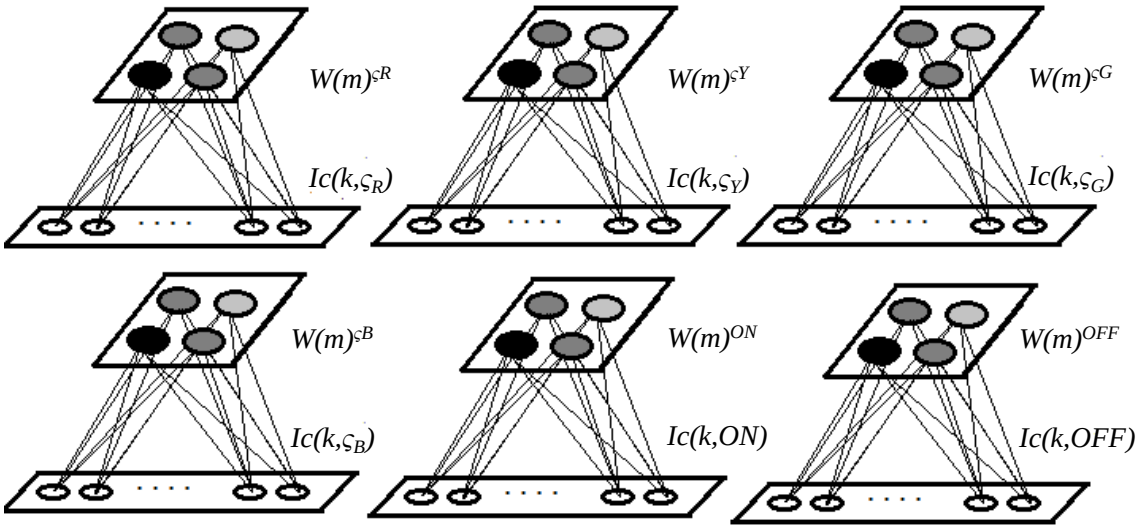


Figura 5.21. Red RESOMV1 que simula la primera capa de V1 y modela el fondo en PBSeg.

Como se vio en la sección anterior, RESOM puede ajustar sus parámetros de aprendizaje para obtener los objetos de fondo y ajustar la sensibilidad de esta red para aprender cambios. En PBSeg se dejaron constantes los parámetros con base a [34] en $\alpha_s=0.42$, $\sigma_s=0.5$, y *RESOMV1* realiza una iteración por cuadro ($N_s=1$). Al igual que el método DR-SOM, en PBSeg se va a generar un arreglo que se actualiza a una neurona por cuadro, pero en este caso no se van a generar imágenes en espacios de color, por tanto, el ordenamiento esta dado por:

$$V_1(k,m)^{z_c,t+1}=\begin{cases}\omega(k,m)^{z_c,n+N_s} & residuo\left(t/M\right)=0\\ V_1(k,m)^{z_c,t} & otra\quad forma\end{cases} \tag{5.29}$$

En los primeros *M* cuadros el arreglo $V_1(k,m)^{z_c,t}$ se actualiza a todas las neuronas por cuadro, pero a partir del cuadro $t=M+1$, la actualización del arreglo se da por la ecuación 5.29. Una vez obtenido el arreglo, el módulo V1 reduce sensibilidad para aprender variaciones como objetos dinámicos o ruido. Al igual que en el capítulo III, si hay un cambio de iluminación entonces las neuronas se volverán a actualizar a todas las neuronas por cuadro en $V_1(k,m)^{z_c}$. El cambio de iluminación se basa en la entropía E debido a su comportamiento ligaritmico que s easemeja a la respuesta del sistema visual humano [124]. De manera similar en DR-SOM con $I_b(x,y,z)^t$, para obtener el modelado de fondo en PBSeg, se obtiene el promedio de $V_1(k,m)^{z_c,t}$ dado por:

$$V_m(k)^{z_c,t} = \frac{1}{M}\sum_m V_1(k,m)^{z_c,t} \tag{5.30}$$

El resultado en $V_m(k)^{z_c,t}$ se transforma a $V_m(x,y)^{z_c,t}$, para obtener seis imágenes que contienen el modelado de fondo, pero descompuesto en los canales RF de color y luminancia. En la Figura 5.22 se puede ver la respuesta de los LGN y la respuesta de $V_m(x,y)^{z_c,t}$, donde se puede apreciar que los LGN responden a todos los objetos de la escena. Sin embargo, los resultados en $V_m(x,y)^{z_c,t}$ no aprenden los objetos dinámicos y son menos sensibles a cambios drásticos en el escenario, debido a RESOMV1 y a la ecuación 5.29. De esta forma, la RESOM ayuda a obtener el fondo para la segmentación de objetos dinámicos.

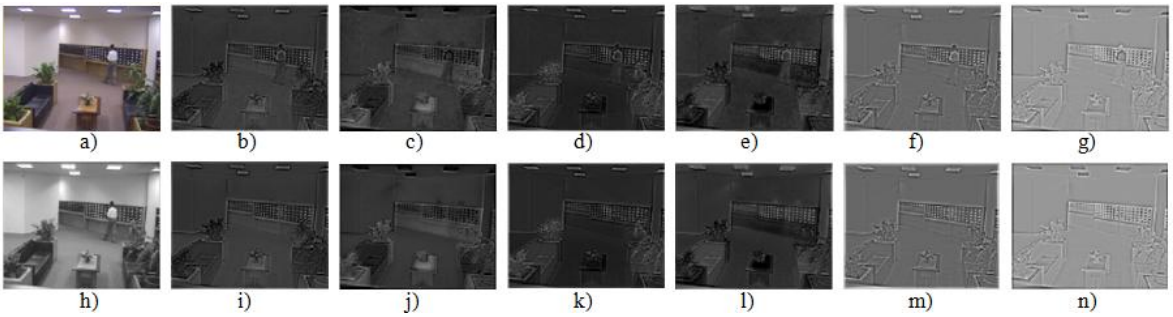


Figura 5.22. Resultados de los *LGN* y *RESOMV1* con $\Theta=-0.2$ y $V_m(x,y)^{zc,t}$. a). Video 'LB' t=170. b). $Ic(x,y,\varsigma_R)$. c). $Ic(x,y,\varsigma_Y)$. d). $Ic(x,y,\varsigma_G)$. e). $Ic(x,y,\varsigma_B)$. f). $Ic(x,y,ON)$. g). $Ic(x,y,OFF)$. h). $I(x,y,V)^t$. i). $V_m(x,y)^{\varsigma R,t}$. j). $V_m(x,y)^{\varsigma Y,t}$. k). $V_m(x,y)^{\varsigma G,t}$. l). $V_m(x,y)^{\varsigma B,t}$. m). $V_m(x,y)^{ON,t}$. n). $V_m(x,y)^{OFF,t}$.

El objetivo de que RESOMV1 elimine ruido y aprenda solo los objetos de fondo, es que las siguientes etapas corticales realicen la segmentación de objetos estáticos. Una vez obtenida la información en los seis canales de $V_m(x,y)^{z_c,t}$, la información se divide en dos:

- Los canales $z_c=\{ \varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B\}$, los cuales contienen la información de color.
- Los canales $z_c=\{ON,OFF\}$ que contienen información de contornos.

El resultado de RESOMV1 es procesado por un conjunto de RF sensibles a OR, lo cual es plausible con los modelos LISSOM, PG-LISSOM, RF-PCNN y las teorías de las columnas verticales.

Campos Receptivos de Orientación de V1 (RFO). Como se vio en el capítulo II y de acuerdo a los modelos LISSOM y SOMRM, en los niveles más bajos de la corteza visual existe un proceso auto-organizado que está ligado a la formación de un mapa de OR que se define mediante el resultado cortical de LISSOM y un conjunto de campos receptivos selectivos a orientación. Por lo tanto, una vez obtenido $V_m(x,y)^{z_c,t}$ en PBSeg, el siguiente paso es mapear esta respuesta mediante un conjunto de campos receptivos que son sensibles a orientación (RFO). Para ello, se definieron un conjunto de campos receptivos dados por:

$$RFO(x,y,\theta)=\exp\left(-\frac{\left[(x-x_c)\cos\theta-(y-y_c)\sin\theta\right]^2}{\sigma_a^2}-\frac{\left[(x-x_c)\cos\theta+(y-y_c)\sin\theta\right]^2}{\sigma_b^2}\right)$$
$$-cRF\exp\left(-\frac{\left[(x-x_c)\cos\theta-(y-y_c)\sin\theta\right]^2}{\sigma_a^2}-\frac{\left[(x-x_c)\cos\theta+(y-y_c)\sin\theta\right]^2}{\sigma_b^2}\right) \quad (5.31)$$

donde θ define la orientación en que el $RFO(x,y)$ es selectivo, cRF es una constante de influencia de la gaussiana alrededor, σ_b, σ_a son los radios a lo largo y a lo ancho respectivamente, (x_c,y_c) es el centro. En la Figura 5.23 se pueden observar ocho RFO selectivos a las direcciones de $\theta=\{0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/4, 7\pi/8\}$, $\sigma_b=2$, $\sigma_a=1$, $cRF=0.35$. El modelo planteado para RFO se basa en el modelo de [34], donde también se utiliza una diferencia de gaussianos pero cada gaussiana es sensible a ciertas orientaciones como se puede observar en la Figura 5.23.

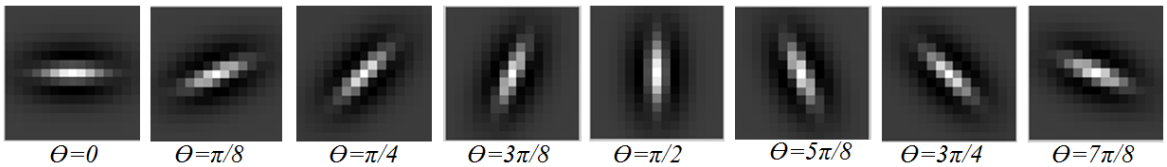


Figura 5.23. Campos Receptivos selectivos a orientación.

Para este trabajo, se plantearon dos tipos de RFO; los RFO para los canales de luminancia $z_c=\{ON,OFF\}$, y los RFO para los canales de color $z_c=\{\varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B\}$. En ambos casos el modelo de la ecuación 5.31 es el mismo y se utilizaron ocho RFO selectivos a las orientaciones de $\theta=\{0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/4, 7\pi/8\}$ para cada canal, lo que cambia es el valor de σ_b, σ_a y *cRF*. Como se menciona en **[**34**,**33], los campos receptivos de V1 sensibles a orientación son células simples, por tanto se pueden definir con una respuesta lineal dada por:

$$V_{nm}(x,y)^{z_c,t} = \sum_{\theta}\left(RFO(x,y,\theta)^{z_c} * V_m(x,y)^{z_c,t} \right) \tag{5.32}$$

$$V_n(x,y)^{z_c,t} = \left(V_{nm}(x,y)^{z_c,t} - \min(V_{nm}(x,y)^{z_c,t}) \right) / \max(V_{nm}(x,y)^{z_c,t}) \tag{5.33}$$

$V_n(x,y)^{z_c,t}$ es una versión normalizada a [0,1] de $V_{nm}(x,y)^{z_c,t}$. En el caso de los RFO de color, estos son utilizados en PBSeg para estimular las regiones de color a las cuales cada canal es selectivo. Como se puede ver en la Figura 5.24b a 5.24e, los canales de color generan patrones de orientación que se pueden estímular con los RFO, y al agregarlas con la sumatoria de la ecuación 5.32, en $V_n(x,y)^{z_c,t}$ se forman regiones que dependen de la distribución de los valores de la respuesta de los canales de color de $V_m(x,y)^{z_c,t}$. Es decir, si la información en un canal de $V_m(x,y)^{z_c,t}$ tiene valores muy homogéneos, entonces se genera una sola región, pero, si los valores son muy irregulares se pueden formar distintas regiones con base a la textura presente en los objetos. Por lo tanto, al ser procesados los canales de color por los RF, se obtiene no solo la información de color seleccionada por cada RF de color, también se obtiene información de la textura de cada objeto en el escenario. Como ejemplo, en $V_m(x,y)^{\zeta R,t}$ la mesa, la repisa y el piso tienen niveles de grises muy similares como se muestra en la Figura 5.24b, pero la salida del RFO de $V_m(x,y)^{\zeta R,t}$, $V_n(x,y)^{\zeta R,t}$, genera diferentes texturas que representan la repisa, el suelo y la mesa. Lo mismo sucede con el resto de los canales.

En el caso de los canales de luminancia, los RFO se utilizan para mejorar el contraste entre los cambios de contraste con orientaciones definidas por los bordes y el resto del escenario. En la Figura 5.25 se puede observar como los bordes están un poco más estimulados en $V_n(x,y)^{\{ON,OFF\}}$ que en $V_m(x,y)^{\{ON,OFF\}}$.

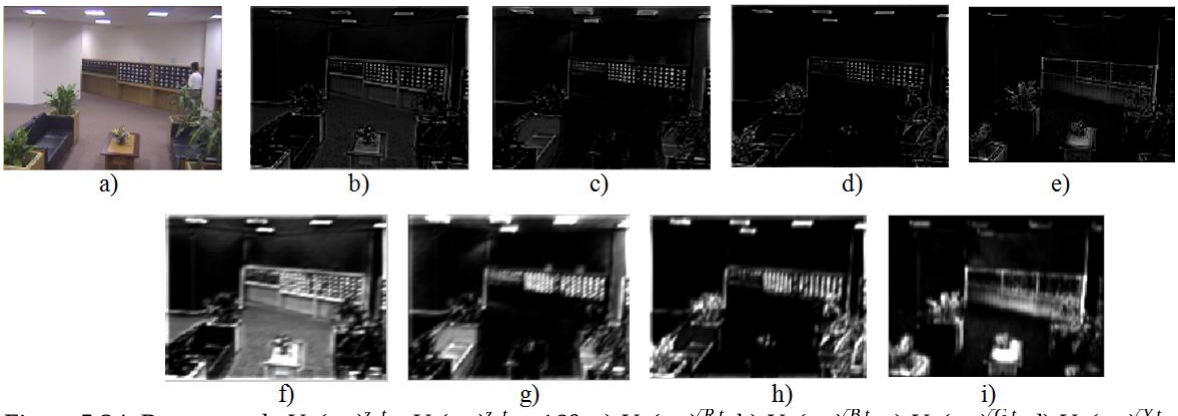


Figura 5.24. Respuesta de $V_m(x,y)^{z_c,t}$ y $V_n(x,y)^{z_c,t}$ $t=160$. a) $V_m(x,y)^{\zeta R,t}$. b) $V_m(x,y)^{\zeta B,t}$. c) $V_m(x,y)^{\zeta G,t}$. d) $V_m(x,y)^{\zeta Y,t}$. e) $V_n(x,y)^{\zeta R,t}$. f) $V_n(x,y)^{\zeta B,t}$. g) $V_n(x,y)^{\zeta G,t}$. h) $V_n(x,y)^{\zeta Y,t}$.

Figura 5.25. Respuesta de $V_m(x,y)^{zc}$ y $V_n(x,y)^{zc}$ en $t=160$. a) $V_m(x,y)^{ON}$. b) $V_n(x,y)^{ON}$. c) $V_m(x,y)^{OFF}$. d) $V_n(x,y)^{OFF}$.

Los resultados en $V_n(x,y)^{z_c,t}$ varían de acuerdo al nivel de influencia de la gaussiana de alrededor caracterizada por *cRFO*. Para $RFO(x,y,\theta)^{\{ON,OFF\}}$ está dado por:

$$cRFO = 0.045 + \mu_d \tag{5.34}$$

De esta forma, la gaussiana de alrededor aumenta su nivel de influencia conforme aumenta la cantidad de ruido causado por fondos dinámicos, μ_d es el valor esperado de $Iv(x,y)^t$. Esto permite eliminar ruido, ya que la *DoG* es menos sensible a dar respuesta a regiones irregulares conforme aumenta la magnitud de la gaussiana de alrededor. En el caso de $RFO(x,y,\theta)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\}}$, *cRFO* se definió en cero para estimular los canales de color y mejorar el contraste de los niveles de grises y mejorar la separación de las regiones basandose en la información espacial.

5.2.4 Procesamiento de color y forma en V1 y V2

Como se puede observar en la Figura 5.13, la formación de las características de color y contornos se va generando en V1 y V2. En los modelos LAMINART y LISSOM pasa algo similar, ya que en el caso de LAMINART el procesamiento de los contornos inicia en V1, y en V2 se realiza el mapeo de dichos contornos en sus respectivas superficies de

profundidad. En PG-LISSOM se enfoca solo al mapeo de bordes en V1. Respecto a color, en TLISSOM se inicia la formación del mapa de CR en V1, y en V2 este mapa genera algunos fenómenos perceptuales. Para ser consistente con estos modelos, PBSeg genera la detección de contornos en V1 con una red neuronal pulsante y en V2 genera el agrupamiento perceptual de los contornos. Para color, PBSeg en V1 solo encuentra un mapa de color y en V2 genera las regiones de color que representan los objetos.

Segmentación de contronos con *Spiking Cortical Model* (SCM): De acuerdo al modelo de LAMINART [41] y diversos modelos relacionados con el funcionamiento de la corteza visual [149], V1 tiene un conjunto de etapas corticales que son sensibles a contornos monoculares y binoculares. Sin embargo, PBSeg trabaja con videos estacionarios adquiridos con una cámara, y por lo tanto se va a descartar información relacionada a la percepción de profundidad. Como se puede ver en el diagrama de LAMINART de la Figura 2.6, V1 contiene dos capas de células simples y una capa de células complejas para el procesamiento de los contornos monoculares. En el modelos PBSeg, la capa RESOMV1 tiene dos niveles de procesamiento de contornos, los cuales corresponden a las redes $W(m)^{ON}$ y $W(m)^{OFF}$ y a las capas de $V_m(x,y)^{ON,t}$ y $V_m(x,y)^{OFF,t}$. Dado que RESOMV1 se basa en las mismas interacciones topográficas que los modelos LISSOM, los cuales modelan el comportamiento de las células simples en V1 [34], entonces $W(m)^{ON}$ y $W(m)^{OFF}$ representan la primer capa de células simples llamada capa 4 de V1. La capa 3B en LAMINART hace una ajuste con la información de la capa anterior, y en PBSeg se representa por $V_n(x,y)^{ON,t}$ y $V_n(x,y)^{OFF,t}$, ya que estas son un promedio de $W(m)^{ON}$ y $W(m)^{OFF}$ y representan un ajuste en la información de la capa anterior que descarta objetos dinámicos.

La tercera capa cortical 2/3, está relacionada con detección monocular y se compone de un conjunto de células complejas que se encargan de completar la formación de los bordes monoculares. En el modelo *spiking* LAMINART [150], la capa 2/3 se modela como una SNN que a partir de la simulación de los potenciales de membrana forma una serie de pulsos cuya codificación en el tiempo forman los bordes monoculares y binoculares. Por tanto, en PBSeg se recurrirá a la simulación de la capa 2/3 con una red pulsante denominada *Spiking Cortical Model* (SCM), la cual es una versión simplificada de la PCNN. A continuación se describirá esta red neuronal.

Spiking Cortical Map (SCM). Como se mencionó en el capítulo II, la PCNN es una red pulsante derivada de los modelos integra y dispara. Esta red genera una sincronización neuronal muy similar a otras redes pulsantes como LEGION y son resultados plausibles con la teoría de la *Gestalt*. Incluso, modelos como PG-LISSOM se valen de variantes de la PCNN para generar la integración de contornos. Una versión simplificada de la PCNN es la *Spiking Cortical Model* (SCM) **[**151**,**152**]**, la cual incluye los mismos módulos que el modelo original, pero a diferencia de la PCNN, el árbol de dendritas esta simplificado de forma tal que el módulo de alimentación esta dado solo por el estímulo de entrada. Derivado de esto, los módulos de alimentación y encadenamiento se pueden fusionar en un solo módulo de actividad interna (*Up*), por tanto, la SCM está definida por el siguiente modelo:

$$Up(i,j)^{n_p} = Up(i,j)^{n_p-1} \cdot \exp\left(-\alpha p_F\right) + I(i,j)\left(1+\beta Yp(i,j)^{n_p-1} * Wp\right) \tag{5.35}$$

$$Yp(i,j)^{n_p} = \begin{cases} 1 & Up(i,j)^{n_p} > Ep(i,j)^{n_p} \\ 0 & otra \quad forma \end{cases} \tag{5.36}$$

$$Ep(i,j)^{n_p} = Ep(i,j)^{n_p-1} \cdot \exp\left(-\alpha p_E\right) + V_E Yp(i,j)^{n_p-1} \tag{5.37}$$

donde *Ep(i,j)* es la función de umbral, *Yp(i,j)* es la salida pulsante, αp_F es el coeficiente de decrecimiento de la actividad interna *Up(i,j)*, αp_E es el coeficiente de decrecimiento de la función de umbral *Ep(i,j)*, β es el parámetro de refuerzo de las neuronas vecinas y V_E es la amplitud de la función de umbral *Ep(i,j)*. En el modelo de [152] existe un parámetro más que es V_L, pero este fue descartado de la ecuación 5.35 ya que no tiene ningún efecto en el módelo de la SCM. Para el caso de PBSeg, $I(i,j)=V_n(x,y)^{z_c,t}$, donde $x=i$, $y=j$, $z_c=\{ON, OFF\}$.

No obstante, un problema que existe en los modelos basados en redes pulsantes es que tienen varios parámetros que son difíciles de sintonizar mediante algún esquema formal y por lo tanto tienen que ser ajustados manualmente. Derivado de esta situación, en [152] se propone una forma de generar un esquema para definir el valor de estos parámetros conforme al contenido de la imagen de entrada. Por lo tanto, se recurrió al método propuesto en [152] para sintonizar los parámetros αp_F, αp_E, β y V_E. En la Figura 5.26 se puede observar la actividad de una neurona donde $\alpha p_F=0.2$, $\alpha p_E=1.1$, $\beta=0.1$, $V_E=10$, el pixel de entrada tiene un valor de $I(i,j)=0.5$, mientras que los vecinos tienen valores de 1/255. Si se analiza el modelo de la SCM, se puede concluir que *Ep(i,j)* es proporcional a

V_E, y a su vez la amplitud de $Ep(i,j)$ es inversamente proporcional a la cantidad de pulsos de salida. Adicionalmente, los pulsos de salida son exponencialmente proporcionales a αp_F y αp_E. β estimula la respuesta de la combinación lineal entre los pesos y los vecinos. En el caso de BPSeg, los pesos se obtuvieron a partir de [152] y están dados por:

$$Wp = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & 0 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix} \tag{5.38}$$

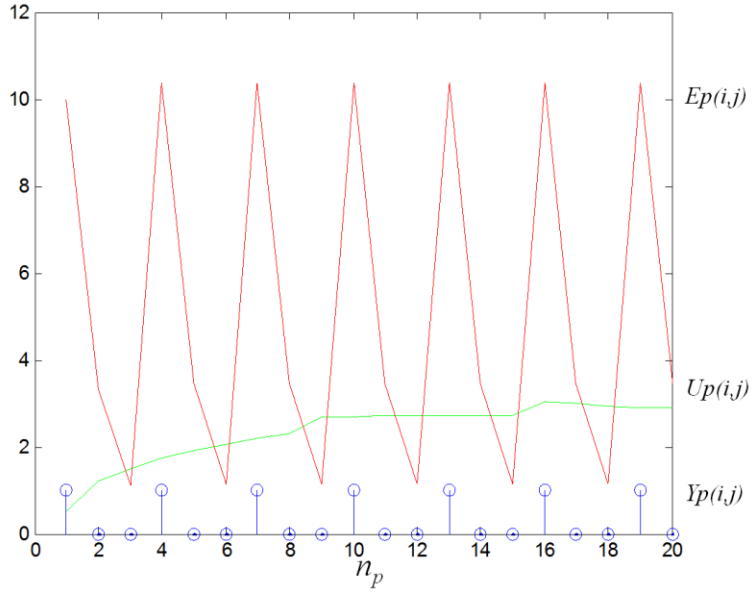


Figura 5.26 Actividad de una neurona. La actividad interna *Up(i,j)* es la señal verde, la función de umbral *Ep(i,j)* es la señal roja y la señal en azul son los pulsos de salida de *Yp(i,j)*.

La respuesta de $V_n(x,y)^{ON,t}$ y $V_n(x,y)^{OFF,t}$ es una segmentación de los contornos como se puede apreciar en la Figura 5.25, donde los valores de los pixeles tienden a comportarse de la siguiente manera:

- Si un pixel (x_1,y_1) no pertenece a un borde, $V_m(x_1,y_1)^{ON,t} \rightarrow 0.5$ y $V_m(x_1,y_1)^{OFF,t} \rightarrow 0.5$.
- Si un pixel (x_1,y_1) pertenece a un borde luz-obscuridad, $V_m(x_1,y_1)^{ON,t} \rightarrow 1$ y $V_m(x_1,y_1)^{OFF,t} \rightarrow 0$.
- Si un pixel (x_1,y_1) pertenece a un borde obscuridad-luz, $V_m(x_1,y_1)^{ON,t} \rightarrow 0$ y $V_m(x_1,y_1)^{OFF,t} \rightarrow 1$.

Con base a estos tres puntos, $V_n(x_1,y_1)^{ON,t}$ y $V_n(x_1,y_1)^{OFF,t}$ forman tres regiones, dos de ellas pertenecientes a los bordes y una al resto de la imagen.

En el análisis en [152], se pudo concluir que la SCM en la primer iteración todas las neuronas pulsan si los pixeles de entrada son diferentes de cero, de manera que $Yp(i,j)^1$ forma una imagen blanca, en la segunda iteración ninguna neurona pulsa, de manera que $Yp(i,j)^2$ forma una imagen negra y en las siguientes iteraciones los pixeles pulsarán formando diferentes regiones en cada iteración. Cómo $V_n(x_1,y_1)^{ON,t}$ y $V_n(x_1,y_1)^{OFF,t}$ tienen sus pixeles diferentes de cero, en $n_p=1$, $Y_p(i,j)$ genera una imagen blanca, en $n_p=2$, $Y_p(i,j)$ genera una imagen negra, y en las siguientes tres iteraciones responderán los pixeles no pertenecientes a bordes, y los pertenecientes a los bordes luz-obscuridad y obscuridad-luz. En consecuencia, las SCM de detección de contornos deben tener cinco iteraciones como se muestra en la Figura 5.27, la primera para que todos los pixeles pulsen (Figura 5.27a), la segunda para que los pixeles no pulsen (Figura 5.27a) y tres para que la SCM detecte las dos regiones de bordes y la región de no bordes.

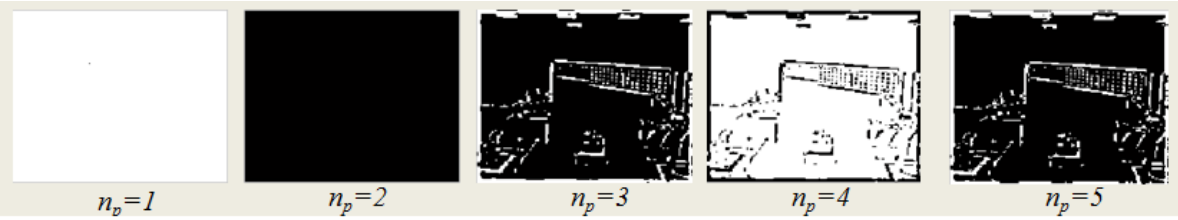


Figura 5.27. Respuesta de $Y_p(i,j)^{n_p}$ en cinco pulsos en el video '*LB*', donde la entrada es $V_n(x,y)^{ON}$. a) $Y_p(i,j)^1$. b) $Y_p(i,j)^2$. c) $Y_p(i,j)^3$. d) a) $Y_p(i,j)^4$. e) a) $Y_p(i,j)^5$.

Este proceso se realiza en cada cuadro con $V_n(x,y)^{ON,t}$ y $V_n(x,y)^{OFF,t}$. De acuerdo a los modelos LAMINART y FORMOTION, los canales ON y OFF combinan su información en V1. En PBSeg, esto sucede de la siguiente manera:

$$\Gamma(i,j)=\sum_{n_p} Y_{ON\,p}(i,j)^{n_p}+\sum_{n_p} Y_{OFF\,p}(i,j)^{n_p},\qquad n_p=3,4,5 \tag{5.39}$$

El resultado contenido en $\Gamma(i,j)$ forma la detección de contornos monocular de V1. Para $Y_{Onp}(i,j)^1$, $Y_{OFFp}(i,j)^1$, $Y_{Onp}(i,j)^2$ y $Y_{OFFp}(i,j)^2$ las neuronas que pulsan no representan bordes. El resultado aunque puede detectar correctamente los bordes, también contiene ruido, por tanto, se realiza un filtrado donde regiones resultantes con áreas menores a 2 pixeles se retiran de $\Gamma(i,j)$. Una vez mejorada $\Gamma(i,j)$, esta se mapea a {0,1} con una función saturación de la siguiente forma:

$$\Gamma_p(\Gamma(i,j)) = \tfrac{1}{2}\left(\left|\Gamma(i,j)\right| - \left|\Gamma(i,j) - 1\right| + 1\right) \tag{5.40}$$

En la Figura 5.28 se pueden observar algunos ejemplos de los resultados obtenidos en la detección de contornos con este esquema basado en SCM. Se puede ver que la segmentación de los bordes es coherente, ya que conservan las formas de los objetos contenidos en la imagen. No obstante, aún cuando se realiza un filtrado de regiones queda un poco de ruido en el resultado de $\Gamma_p(i,j)$. Con este módulo de detección de contornos se completa una de las principales tareas en V1 que es la detección de forma mediante los contornos.

a) b) c) d) e) f)

Figura 5.28. Resultados de $\Gamma_p(i,j)$. a) Video '*LB*', *t=160*. b) $\Gamma_p(i,j)$ de a). c). Video '*LS*', *t=50*. d) $\Gamma_p(i,j)$ de c). e) Video '*WS*', *t=100*. f) $\Gamma_p(i,j)$ de e).

Agrupamiento perceptual en V2. Una vez que se obtienen los contornos de los objetos el siguiente paso es el agrupamiento perceptual para detectar la forma de los objetos, lo cual lo realiza el módulo V2. Cómo se mencionó anteriormente, PBSeg utiliza como entrada videos de cámaras estacionarias, por lo que no se va a considerar el análisis de profundidad ni el mapa de DO.

En psicología, agrupamiento perceptual se refiere a la detección de formas de objetos a partir de la integración de contornos y un conocimiento previo. En el caso de neurociencias computacionales, el agrupamiento perceptual es resultado de la formación de grupos de neuronas a través de la sincronización de oscilaciones y pulsos. Algunos modelos que formalizan esta etapa del proceso de percepción son *Perceptual Grouping LISSOM* (PG-LISSOM) [34], las redes LEGION [46] y *Wilson Cowan* [47]. Pare este caso, dado que ya se tiene un resultado de detección de contornos con SCM en V1, solo falta definir un enfoque cognitivo; es decir, que V2 utilice un conocimiento previo. Este conocimiento previo se define mediante la siguiente relación:

$$\Gamma_c(i,j) = \Gamma_c(i,j) + \Gamma_p(i,j) \tag{5.41}$$

$$\Gamma_q(\Gamma_c(i,j)) = \tfrac{1}{2}\left(\left|\Gamma_c(i,j)\right| - \left|\Gamma_c(i,j) - 1\right| + 1\right) \tag{5.42}$$

En este caso, $\Gamma_c(i,j)$ se actualiza en cada cuadro y su condición inicial es ceros. Conforme pasan los cuadros, los bordes de los objetos estáticos se mantienen en $\Gamma_q(i,j)$, mientras que los bordes de los objetos dinámicos y ruido se van inhibiendo ya que no se repiten en los resultados de $\Gamma_c(i,j)$. En la Figura 5.29 se puede observar el resultado del agrupamiento perceptual en varios cuadros que pueden contener objetos dinámicos y ruido causado por fondos dinámicos, pero que no afectan a $\Gamma_q(i,j)$. El agrupamiento perceptual en $\Gamma_q(i,j)$ genera resultados muy similares a $\Gamma_p(i,j)$ cuando el video tiene escenarios en condiciones normales como en los videos 'LB' y 'LS', pero en el caso de escenarios con fondos dinámicos u objetos dinámicos que se mantienen estacionarios los resultados difieren entre $\Gamma_q(i,j)$ y $\Gamma_p(i,j)$, de tal manera que $\Gamma_q(i,j)$ elimina información que no es del fondo. $\Gamma_q(i,j)$ define el agrupamiento perceptual en PBSeg ya que contiene información de las formas y contornos de los objetos de fondo con base a la sincronización de pulsos de una red neuronal en V1 y la integración de un conocimiento previo. Este conocimiento se basa en la acumulación de información en $\Gamma_q(i,j)$ de contornos de cuadros pasados, eliminando detalles no relevantes en la percepción de formas.

a) b) c) d) e) f)

Figura 5.29. Resultados de Agrupamiento perceptual en $\Gamma_q(i,j)$. a) Video 'WS', t=550. b) $\Gamma_q(i,j)$ de a). c) Video 'BT', t=550. d) $\Gamma_q(i,j)$ de c). e) Video 'SS', t=550. f) $\Gamma_q(i,j)$ de e).

Segmentación de color con SCM y mapas de color en V1 y V2. De acuerdo a [143], el objetivo de TLISSOM es estimular un conjunto de neuronas sensibles a color (CR) que funcionan como un mapa selectivo que indica la distribución de color en el estímulo visual. Este conjunto de neuronas tienen un comportamiento que se puede asemejar a un mapa de matiz *H* [143], además estas neuronas interactúan con otras neuronas sensibles a otras características como dominancia ocular (DO) y orientación (OR). En la Figura 5.30 se puede observar un ejemplo donde el mapa de color CR interactúa con otros mapas de características. El mapa selectivo de CR a su vez estimula el mapa selectivo de OR mediante un proceso de retinotopia. Con base a este funcionamiento se puede asumir lo siguiente:

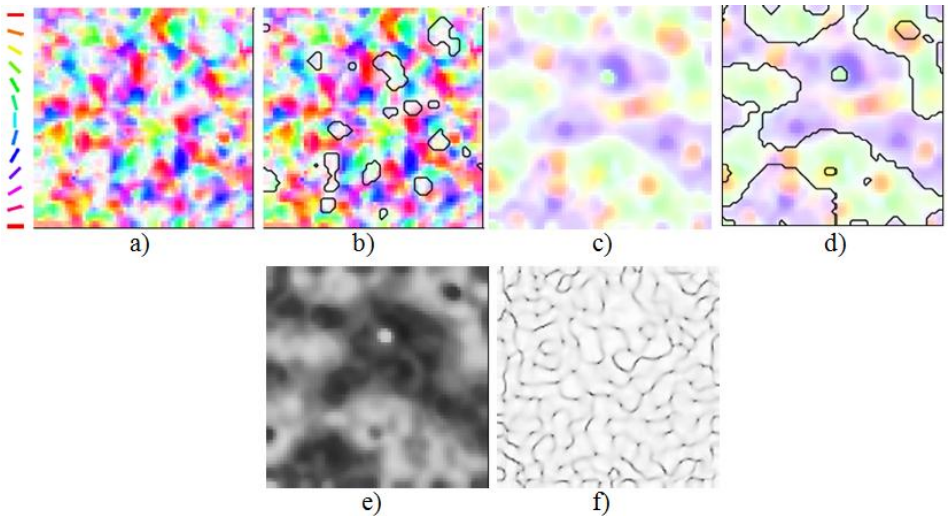


Figura 5.30. Mapas en V1 y V2. Mapas (a) a c). fueron obtenidos de **[**41**]**, el mapa en f) fue obtenido de **[**34**]**. a). Mapa de OR. b). Selección de regiones a partir de CR. c). Mapa de CR. d). Selección de regiones a partir de DO. f). Mapa de gradiente.

1. En V1 los mapas de características OR, CR, DO, gradiente y frecuencia espacial, se van modelando a partir del estímulo visual durante periodos de aprendizaje. Después de dichos periodos, se forman mapas selectivos que estimulan ciertos patrones de orientación, color, disparidad, gradiente y frecuencia a partir del estimulo visual actual y los patrones aprendidos.
2. Los mapas selectivos de OR, CR, DO, gradiente y frecuencia espacial se retroalimentan entre sí para definir el estado del color, orientación, disparidad, gradiente y frecuencia de los objetos presentes en la escena. Esta información pasa a V4 para iniciar con el proceso de percepción.

De esta forma, el mapa CR juega un papel importante en los procesos de percepción ya que estimula los elementos más relevantes del mapa OR y está relacionado con el mapa de gradiente y DO. Esta información influye en otras capas corticales para definir procesos de atención y cognición visual. En términos de procesamiento de imágenes, el mapa de neuronas selectivo a color funciona de forma similar a la información de matiz del espacio HSV, incluso en [143] se estudian estas neuronas mediante un histograma que representa la distribución de matiz generada por la información de color presente en el estimulo de entrada. Sin embargo, la diferencia más significativa entre la información H y el mapa CR está en que este último se forma a partir del procesamiento de la información de los campos receptivos de color. PBSeg desarrolla su mapa de color a partir de la información de matiz

y la información generada por la información de color presente en $V_m(x,y)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\},t}$. Para realizar este mapa, primero se revisa la distribución de color revisando la dominancia en los niveles de color en $V_m(x,y)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\},t}$ generando un cuadro $I_{rb}(x,y)^t$ dado por:

$$I_{rb}(x,y)^t = \begin{cases} 0.25 \;\; si \;\; \max(V_m(x,y)^{zc}) \;\; esta \;\; en \;\; el \;\; canal \;\; \varsigma_Y \\ 0.5 \;\; si \;\; \max(V_m(x,y)^{zc}) \;\; esta \;\; en \;\; el \;\; canal \;\; \varsigma_G \\ 0.75 \;\; si \;\; \max(V_m(x,y)^{zc}) \;\; esta \;\; en \;\; el \;\; canal \;\; \varsigma_B \\ 1 \;\; si \;\; \max(V_m(x,y)^{zc}) \;\; esta \;\; en \;\; el \;\; canal \;\; \varsigma_R \end{cases} \tag{5.43}$$

En la Figura 5.31a se observa el resultado de $I_{rb}(x,y)^t$ del video 'LB' y el histograma resultante hI_{rb} el cual indica la distribución de color dado por los campos receptivos de color. En ocasiones, esta distribución puede ser engañosa como se observa en la Figura 5.31c donde un cuadro con una distribución mayoritariamente azul genera un resultado que muestra un hI_{rg} en el que el color dominante es amarillo y también se muestra un poco de verde. Este error sucede porque el video tiene baja saturación y el color dominante es cyan. Este color cyan con baja saturación debería mapearse para el lado del campo receptivo del azul, no obstante, debido a la baja saturación del video '*WS*', los LGN mapean este color como verde y amarillo. Este efecto sucede porque en cada RF, la información es variante a H y S, de manera que un color mal saturado se puede confundir con un color de matiz menor. En un análisis de la respuesta de los RF de color, se observó que un color con saturación muy baja va a tener un matiz que se va a confundir con otro matiz que esta desfasado aproximadamente en π/4.

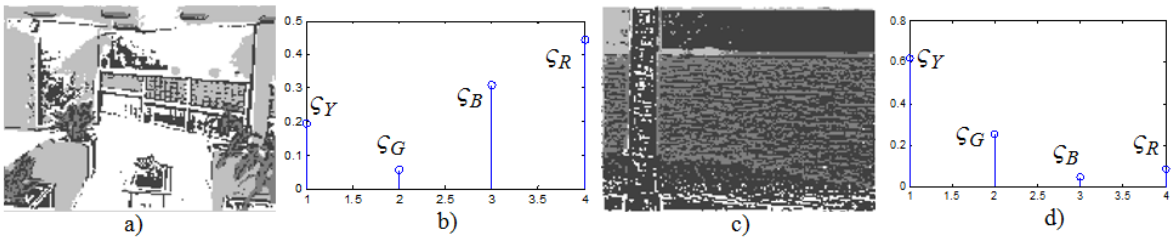


Figura 5.31. Cuadro $I_{rb}(x,y)^t$ y el histograma. a) $I_{rb}(x,y)^t$ en video '*LB*'. b) Histograma hI_{rb} de a). c) $I_{rb}(x,y)^t$ en video '*WS*'. d) Histograma hI_{rb} de c).

De esta manera, se puede notar que el histograma hI_{rb} no siempre es confiable para indicar a módulos posteriores de PBSeg la distribución de color en $V_m(x,y)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\}}$. Sin embargo, esto sucede porque la información de matiz en $V_m(x,y)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\}}$ puede estar desfasada si la saturación del color es baja. Por lo tanto, una forma de realizar un mapa de

color es utilizar la información de matiz H pero analizando qué tanto puede ser similar esta información a hI_{rb} con el objetivo de conocer si la información en $V_m(x,y)^{\{\varsigma R,\varsigma Y,\varsigma G,\varsigma B\}}$ esta desfasada. Para ello, se debe tomar en cuenta qué colores con matices amarillos, cyan y magenta están mapeados en dos canales de $\{\varsigma_R,\varsigma_Y,\varsigma_G,\varsigma_B\}$, por lo tanto, es posible que una distribución de matiz H rotado en $\pi/4$ tenga más similitud a hI_{rb}. En consecuencia, se genera un mapa de matiz H_3 dado por:

$$\begin{aligned} & I(x,y,H_3) = I(x,y,H) + \pi/4 \\ & si \quad I(x,y,H) > 2\pi \\ & \quad I(x,y,H_3) = I(x,y,H) - 2\pi \end{aligned} \tag{5.44}$$

H_3 genera un patrón que puede asemejarse más al patrón de hI_{rb} si la saturación en un video es muy baja. Por lo tanto, para generar un mapa de color basado en matiz como en [**143**], se forma un mapa o histograma de color a partir de:

$$hCr = \begin{cases} hI_{rb} & si \quad \sum_q \left|hI_h(q) - hI_{rb}(q)\right| \le \sum_q \left|hI_h(q) - hI_r(q)\right| \\ hI_r & si \quad \sum_q \left|hI_h(q) - hI_{rb}(q)\right| > \sum_q \left|hI_h(q) - hI_r(q)\right| \end{cases} \tag{5.45}$$

donde hI_h es un histograma de matiz H dividido en cuatro secciones que representan el contenido de amarillo ($q=1$), verde ($q=2$), azul ($q=3$) y rojo ($q=4$) $q=1,..,4$, hI_r es el histograma de H_3 el cual esta dividido en cuatro secciones, las cuales son también amarillo ($q=1$), verde ($q=2$), azul ($q=3$) y rojo ($q=4$). En la siguiente subsección se verá como este mapa de color *hCr* puede ser utilizado para el proceso de percepción de objetos basándose en color.

Etapa cortical de V2 basada en SCM. El mapa de color *hCr* que se obtiene en V1 puede ayudar a definir la distribución de color en los canales de color de $V_m(x,y)^{z_c,t}$, pero no realiza el proceso de segmentación de objetos. Este proceso inicia en V2, donde una red variante de SCM que se denominó Modelo Cortical Pulsante de la Gestalt (GSCM) genera regiones basándose en el color y la textura de los objetos. Esta variante es una SCM que se aplica en cada uno de los canales de color de $V_n(x,y)^{z_c,t}$ pero tiene como parámetros la caída de la actividad interna αq_F, la caída de la función de umbral αq_E y la influencia del vecindario σ_{zc}. El modelo propuesto esta dado por:

$$U_{z_c}(i,j)^{n_z} = U_{n_c}(i,j)^{n_z-1} \cdot \exp(-\alpha_F z) + V_n(i,j)^{n_c}\left(1 + Y_{z_c}(i,j)^{n_z-1} * W_{z_c}\right) \quad (5.46)$$

$$Y_{z_c}(i,j)^{n_z} = \begin{cases} 1 & U_{z_c}(i,j)^{n_z} > E_{z_c}(i,j)^{n_z} \\ 0 & \textit{otra forma} \end{cases} \quad (5.47)$$

$$E_{z_c}(i,j)^{n_z} = E_{z_c}(i,j)^{n_z-1} \cdot \exp(-\alpha_E z) + Y_{z_c}(i,j)^{n_z-1} \quad (5.48)$$

$$W_{z_c} = w_{z_c}(p,q) = \exp\left(-\frac{(p-i)^2 + (q-j)^2}{\sigma_{z_c}^2}\right) - w(i,j) \qquad (p,q) \in N_{z_c}(i,j) \quad (5.49)$$

donde n_z es el índice de iteraciones de la GSCM, $z_c=\{\varsigma_R, \varsigma_G, \varsigma_B, \varsigma_Y\}$, $U_{zc}(i,j)^{n_z}$ es la actividad interna de la GSCM en el canal z_c, $E_{zc}(i,j)^{n_z}$ es la función de umbral en el canal z_c, $Y_{zc}(i,j)^{n_z}$ es la salida de la GSCM en el canal z_c y N_{zc} es el vecindario del pixel (i,j) en el canal z_c, W_{zc} es el kernel de pesos en el canal z_c. $\alpha_E z$ y $\alpha_F z$ definen la caída de las funciones de umbral y actividad interna respectivamente. La red tiene cuatro canales $z_c=\{\varsigma_R, \varsigma_G, \varsigma_B, \varsigma_Y\}$ como se muestra en la Figura 5.32 donde $n_z=1,2,3$. Con base a [152], se estableció que para toda la red $\alpha_F z=0.2$ y $\alpha_E z=0.7$, además, se estableció que la cantidad máxima (NZ) de iteraciones n_z en condiciones de iluminación normal debe ser de NZ =3. Por tanto, GSCM define su comportamiento con base a la cantidad de vecinos del pixel (i,j), ya que casi todos los parámetros de la red se hicieron constantes excepto σ_{zc}. En GSCM cada canal z_c tiene su parámetro σ_{zc}, pero todos los canales tienen el mismo valor en NZ, $\alpha_F z$, $\alpha_E z$. De esta manera, la GSCM queda solo en función de σ_{zc}, el cual va a indicar el tamaño del vecindario $N_{zc}(p,q)$ del pixel (i,j) de cada canal de color.

Este parámetro puede ser útil en el proceso de segmentación perceptual de objetos, ya que al aumentar el valor de σ_{zc}, aumenta el tamaño de las regiones generadas, lo cual se puede apreciar en la Figura 5.33 donde se muestra el resultado de $Y_{zc}(i,j)^3$ con distintos valores de σ_{zc}. Como se puede ver en las Figuras 5.33b y 5.33c, los objetos de tamaño pequeño respecto a la escena, son representados más coherentemente con valores de σ_{zc} bajos, mientras que los objetos de tamaño grande con respecto a la escena, son representados más coherentemente con valores de σ_{zc} altos. Esto sucede porque la GSCM puede generar grupos con más información de un vecindario de un pixel $p_e(x,y) \epsilon V_n(x,y)^{\{\varsigma R, \varsigma Y, \varsigma G, \varsigma B\}}$, ya que W_{zc} forma una gaussiana que aumenta su radio y amplitud al aumentar el tamaño de σ_{zc} como se ve en la Figura 5.34. En consecuencia, al aumentar el

tamaño de W_{zc} aumenta el amaño del vecindario N_{zc}, causando regiones de mayor tamaño en $Y_{zc}(i,j)^3$.

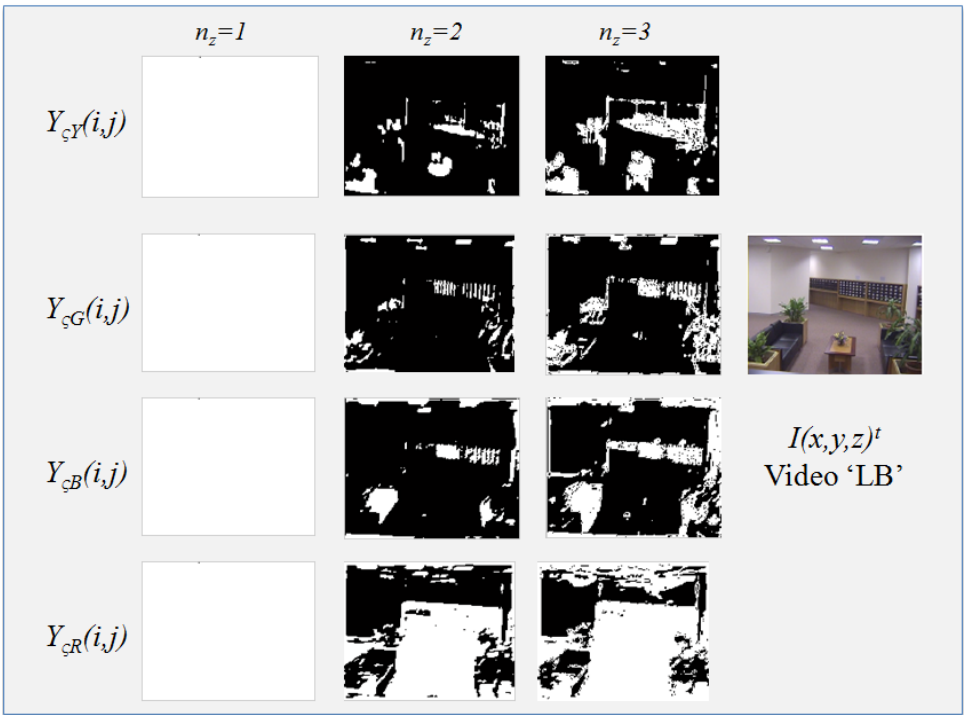


Figura 5.32. Pulsos donde $n_z=1,2,3$ de GSCM.

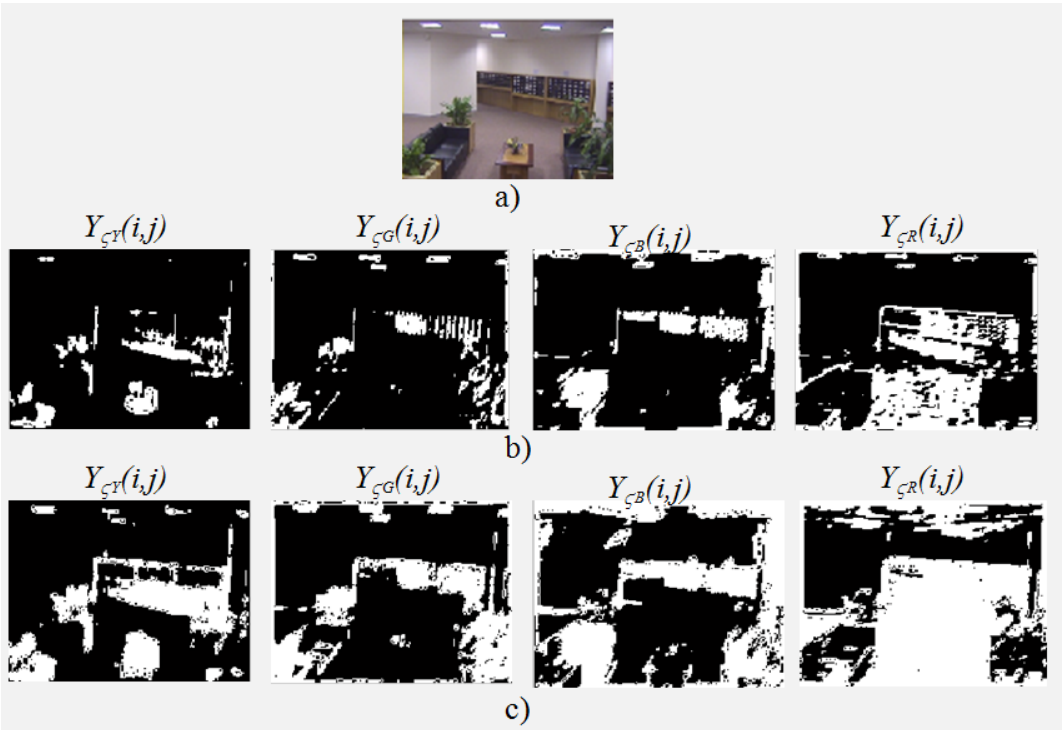
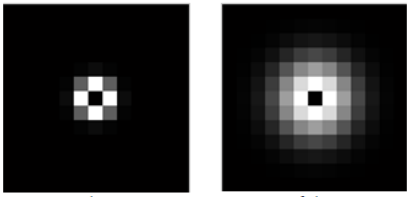


Figura 5.33. Salida de $Y_{zc}(i,j)$ en $t_z=3$. (a) Video '*LB*', t=10. (b) $Y_{zc}(i,j)^{tz}$, en $t_z=3$ y $\sigma_{zc}=0.5$. (c) $Y_{zc}(i,j)^{tz}$, en $t_z=3$ y $\sigma_{zc}=2$.



a) b)

Figura 5.34. Pesos de Wz_c si (a) σ_{zc}=0.5. (b) σ_{zc}=2.

Aspectos bioinspirados de la manipulación de los pesos en W_{zc}: El manejo del tamaño de σ_{zc} es plausible con principios perceptuales, ya que de acuerdo a la red de conexiones difusivas [57] que se vio en el capítulo II, se menciona que el principio de proximidad de la *Gestalt* sugiere que debe haber una fuerte conexión entre osciladores que corresponden a puntos próximos. Este acoplamiento es adaptivo y se puede dar por difusiones como el de la ecuación 2.9 o se puede dar entre osciladores con base a:

$$\Xi_{ij} = \begin{cases} \exp\left(-\left| o_i - o_j \right|^2 \Big/ \beta_o^2 \right) & si \quad j \in N_i \\ 0 & otra \quad forma \end{cases} \tag{5.50}$$

donde (o_i,o_j) son las posiciones de los osciladores, N_i es el vecindario del oscilador de la posición o_i, β_o sintoniza la capacidad de acoplamiento entre osciladores, el cual conforme aumenta su valor, serán más los osciladores que se van a agrupar. Tanto la ecuación 2.9 como la ecuación 5.50 tienen un efecto similar a la ecuación 5.49 ya que el acoplamiento entre osciladores es similar a aumentar el vecindario en W_{zc}.

En [153], se propone un modelo de campos receptivos de color sensibles a longitudes de onda rojo, verdes, amarillo y azul definidos mediante diferencias de gaussianos similares a los propuestos a los módulos LGN. Dichos campos receptivos tienen un área de influencia que depende de la cantidad de color presente en la imagen original. Es decir, entre más grande sea una superficie de un color determinado, mayor es la amplitud del campo receptivo, lo cual resulta útil en segmentación por textura.

Adicionalmente, si se analizan los casos de **[**153**,**57**,**152] se puede notar que tanto capas corticales oscilantes como en los campos receptivos, existe la posibilidad de obtener regiones más grandes en la segmentación y en detectar los diferentes tipos de textura presentes en la imagen al aumentar la influencia de los datos de neuronas vecinas. De

acuerdo a [149], en V2 existe una sintonización de ciertas neuronas para adaptarse al comportamiento de la textura de los patrones visuales de entrada. Con base a esto, en PBSeg se utiliza un mecanismo similar al principio de proximidad de la *Gestalt* como en **[**57**,**153**]** ya que en este caso un conjunto de neuronas pulsantes dadas por la GSCM van a generar regiones que dependen de la sincronización de los pulsos de la GSCM y del parámetro σ_{zc}. Este parámetro depende de la distribución de los niveles de grises en los campos receptivos de color, tal como en **[**153**]**, donde al aumentar la cantidad de color aumenta el tamaño W_{zc}. El aumentar o disminuir el tamaño de W_{zc} en función de la distribución de niveles de grises genera resultados variantes no solo a color, también a textura. En el caso de PBSeg, el valor σ_{zc} depende del valor de *hCr*. Este valor se determina en tres pasos:

1. En el primer paso, PBSeg sugiere un valor de σ_{zc} para cada canal de $Yz_c(i,j)^{n_z}$. Para tener plausibilidad con lo que se propone en **[**153**]**, el tamaño de Wz_c estará en función de *hCr*, ya que este brinda información de cómo está distribuido el color en $I(x,y,z)^t$ y $V_n(x,y)^{z_c,t}$. En este caso, al aumentar la cantidad de pixeles diferentes de cero en $V_n(x,y)^{z_c,t}$, debe aumentar el tamaño de W_{zc}. Por tanto, los valores de σ_{zc} están determinados por:

$$\sigma_{z_c} = \sigma o_{z_c} + 1 + \frac{1}{1+\exp\left(-30\left(hCrN - 0.55\right)\right)} \tag{5.51}$$

En el tercer paso se discute el valor de σo_{zc}, el cual en condiciones iniciales es cero, *hCrN* es *hCr* normalizado respecto a la cantidad total de pixeles *K*. El siguiente término en la ecuación 5.51 es para que el valor mínimo en σ_{zc} sea uno, ya que valores menores a uno en σ_{zc} los objetos pequeños del escenario generan regiones separadas similares a ruido. El tercer término en la ecuación 5.51 tiene un comportamiento sigmoidal y está sintonizado para que el valor más alto de σ_{zc} sea 2, ya que valores más altos hacen que en $n_z=3$ las regiones entre objetos separados se empalmen.

2. Una vez estimado el valor de σ_{zc}, se obtiene $Y_{zc}(i,j)^{n_z}$ con $NZ=3$, y tal como se mencionó en el punto pasado, en condiciones iniciales, $\sigma o_{zc}=\{0,0,0,0\}$, por lo tanto, en el primer cuadro se obtiene un resultado que está en términos solo de la distribución de color dada por *hCrN*.

En la Figura 5.35 se ven dos ejemplos de esta estimación de $Y_{zc}(i,j)^{n_z}$ en $t=1$ cuando $\sigma o_{zc}=\{0,0,0,0\}$. Aparte de mostrar en la Figura 5.35 la información resultante de $Y_{zc}(i,j)^{n_z}$ en $t=1$, se muestra que el canal $Y_{\varsigma G}(i,j)^3$ forma una región que corresponde a la superficie de mar del video '*WS*', mientras que el canal $Y_{\varsigma B}(i,j)^3$ tiene poca información. Si se observa el cuadro del video '*WS*' se esperaría que el canal $Y_{\varsigma B}(i,j)^3$ fuera el que contuviera la información del mar y como se observa en la Figura 5.36a la información de matiz H del cuadro es principalmente azul. No obstante, como se mencionó anteriormente la baja saturación del cuadro hace que la información en $V_n(x,y)^{zc,t}$ se mapee en niveles de matiz más bajos (se debe tomar en cuenta que la información de los RF de color es sensible a H y S, por lo que la baja saturación genera niveles de grises similares a matices más bajos en un RF de color,

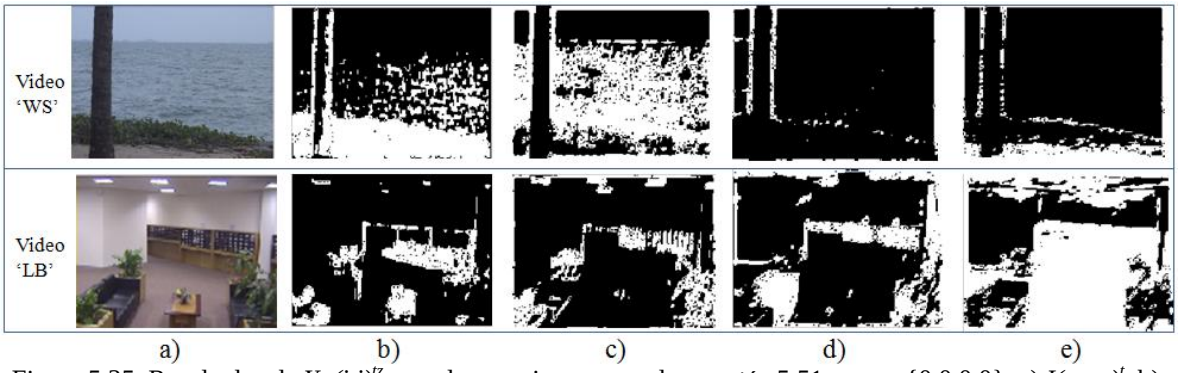


Figura 5.35. Resultados de $Y_{zc}(i,j)^{tz}$ cuando se estima σ_{zc} con la ecuación 5.51 y $\sigma o_{zc}=\{0,0,0,0\}$. a) $I(x,y,z)^t$. b) $Y_{\varsigma Y}(i,j)^3$. c) $Y_{\varsigma G}(i,j)^3$. d) $Y_{\varsigma B}(i,j)^3$. e) $Y_{\varsigma R}(i,j)^3$.

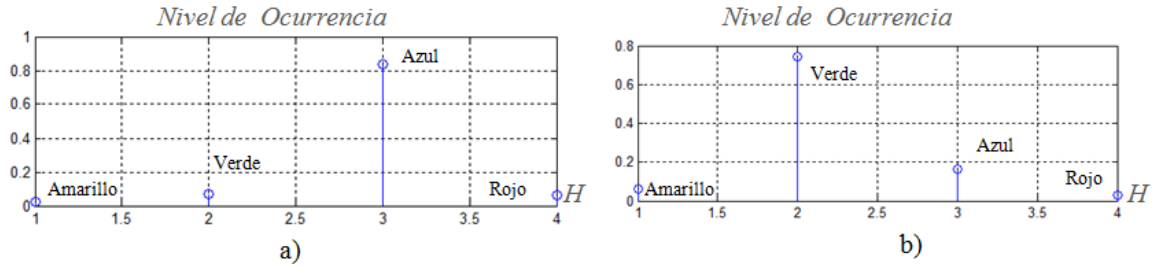


Figura 5.36. Mapas de color basados en matiz del video 'WS'. a) Distribución de color basado en H (hI_h). b) Distribución de color basado en H (hI_r), el cual de acuerdo a la ecuación 5.45 es *hCr*.

Tal como se ve en las Figuras 5.19). Por lo tanto, para el caso del video '*WS*' el mapa de color *hCr* está dado por H_3, el cual se muestra en la Figura 5.36b. En el caso del video '*LB*', los colores en el escenario tienen niveles de saturación mayores que en el video '*WS*'. De esta manera, al aplicar la ecuación 5.44, el mapa de *hCr* utilizado para definir el tamaño de Wz_c es el mismo que hI_h.

3. Una vez obtenida la primera estimación de $Y_{zc}(i,j)^3$, PBSeg analiza si el valor de σ_{zc} propuesto por *hCr* y la función sigmoide en la ecuación 5.51 son los apropiados para definir el tamaño de Wz_c que defina regiones que representen de forma más coherentemente posible los objetos en el fondo.

El valor de σ_{zc} genera un tamaño inicial de Wz_c que puede fallar si existe más de un objeto mostrado en un canal de $V_n(x,y)^{zc,t}$. Por ejemplo, si un escenario tiene varios objetos rojos, estos pueden generar un nivel muy alto en *hCr(4)*, y hacer que el valor de $\sigma_{\varsigma R}$ sea muy alto, causando que las regiones de los objetos sean más grandes que el objeto, lo cual genera una pérdida de coherencia en la segmentación. Por lo tanto, es necesario verificar si un canal de $V_n(x,y)^{zc}$ tiene más de un objeto, y si es así, reducir el tamaño de σ_{zc}, para lo cual se utiliza σo_{zc}. PBSeg tiene un algoritmo que utiliza el resultado de $Y_{zc}(i,j)^3$ para conocer cuantas regiones se forman y con base a esto verificar si es necesario reducir o aumentar el valor de σ_{zc}. Para realizar esto, primero se realiza un etiquetado o *Labeling* a $Y_{zc}(i,j)^3$ con conectividad 8, $Y_{8zc}(i,j)$. Luego, se encuentra la cantidad de regiones en cada canal de $Y_{8zc}(i,j)^{t_z}$ mediante lo siguiente:

$$L\_Max(z_c) = \max(Y_{8z_c}(i,j)) \tag{5.52}$$

En el arreglo $L\_Max(z_c)$ está contenida la cantidad de regiones en cada canal de $Y_{zc}(i,j)^{n_z}$, y a su vez esta información está relacionada con la cantidad de objetos presentes en cada canal de $Y_{zc}(i,j)^{n_z}$. Por lo tanto, si en un canal cualquiera ς_x, $L\_Max(\varsigma_x)=1$, entonces existe solo un objeto en este canal. Si en un canal cualquiera ς_x, $L\_Max(\varsigma_x)>1$, entonces es necesario verificar si son varios objetos o existe ruido. Para ello, se busca el área de la región de mayor tamaño en cada canal de $Y_{8zc}(i,j)^{n_z}$ y se compara con la cantidad total de pixeles diferentes de cero en cada canal de la siguiente manera:

$$A\_Blob(r_d, z_c) = \sum_{i \in R_g} \sum_{j \in R_g} Y_{8z_c}(i,j) \tag{5.53}$$

$$A\_Max(z_c) = \max\left[A\_Blob(r_d, z_c)\right] \tag{5.54}$$

$$ObjArea(z_c) = \frac{A\_Max(z_c)}{\sum_i \sum_j Y_{z_c}(i,j)^{n_z}} \tag{5.55}$$

Primero, de acuerdo a la ecuación 5.53, se encuentra el área de todas las regiones definidas en todos los canales de $Y_{8zc}(i,j)^{n_z}$, donde $r_d=1,\ldots, L\_Max(z_c)$. Segundo, se busca la

región de mayor tamaño en cada canal, de forma que el resultado en la ecuación 5.54 contiene el área de las regiones más grandes en cada canal de $Y_{8zc}(i,j)^{n_z}$. Finalmente, cada área definida en $A\_Max(z_c)$ se divide entre la cantidad total de pixeles con valor de uno en $Y_{zc}(i,j)^{n_z}$. Si este cociente es muy cercano a uno en un canal $z_c=\varsigma_x$, entonces es muy posible que en ese canal $z_c=\varsigma_x$ solo este definido un objeto, mientras que el resto de las regiones son ruido u objetos poco significativos en la imagen. Experimentalmente, se determinó que si $ObjArea(\varsigma_x)>0.85$, entonces está definido un objeto, mientras que el resto de las regiones son ruido u objetos poco significativos en la imagen. Por tanto, si $ObjArea(\varsigma_x)\leq 0.85$, entonces $Y_{\varsigma x}(i,j)^{n_z}$ contiene regiones que representan varios objetos.

Basándose en las condiciones descritas, PBSeg considera que existe un solo objeto en un canal cualquiera $z_c=\varsigma_x$ si $L\_Max(\varsigma_x)=1$ o $ObjArea(\varsigma_x)>0.85$. Este análisis se realiza si el tamaño de Wz_c es lo suficientemente grande para integrar objetos que en realidad están separados en la escena al menos por un pixel; esta condición sucede cuando $\sigma_{zc}>1.5$. En caso de que se cumpla esta condición, es necesario analizar si existe un solo objeto en el escenario o varios. Si existe un solo objeto en un canal cualquiera $z_c=\varsigma_x$, el valor de $\sigma_{\varsigma x}$ se incrementa para que exista más coherencia entre el objeto y la región generada incrementando el valor de $\sigma o_{\varsigma x}$. Si existen varios objetos en el escenario, entonces se realiza un decremento al valor de $\sigma o_{\varsigma x}$ para evitar problemas de integración de regiones de objetos que en realidad se encuentran separados. Esto se puede ver en forma más general mediante lo siguiente:

$$\sigma o_{z_c}=\begin{cases} 0.5 & Si \quad \left(\sigma_{z_c}\geq 1.6 \,\&\, \left(L\_Max(z_c)=1 \,|\, ObjArea(z_c)>0.85\right)\right) \\ -0.5 & Si \quad \left(\sigma_{z_c}\geq 1.6 \,\&\, \left(L\_Max(z_c)>1 \,\&\, ObjArea(z_c)\leq 0.85\right)\right) \\ 0 & otra \quad forma \end{cases} \tag{5.56}$$

En la Figura 5.37, se pueden ver los resultados en el video '*LB*' y '*WS*' después que se realizó el ajuste de los valores de σo_{zc}. En los videos de '*WS*' y '*LB*', existe un canal donde se realizó el análisis de tamaño de σ_{zc}. En el caso del video '*WS*', $Y_{\varsigma G}(i,j)^3$ tuvo un ajuste de $\sigma o_{\varsigma G}=0.5$, haciendo que el mar sea representado de forma más coherente que en el caso de la Figura5.35b. En el caso del video '*LB*', $Y_{\varsigma R}(i,j)^3$ tuvo un ajuste de manera que $\sigma o_{\varsigma R}=-0.5$, lo cual hace que se genere un poco más de ruido en el resultado de las regiones (ver Figura 5.35e), pero se eliminan regiones que se generan por los reflejos de la

iluminación. Como en el ejemplo del video '*LB*', existen muchas regiones que contienen ruido, el cual consiste en puntos negros dentro de la región. Este ruido se presenta en todos los videos y es causado por la misma variación existente en la secuencia de video. Para eliminarlo, se utiliza un arreglo de resultados de $Y_{zc}(i,j)^{n_z}$, el cual se describe a continuación.

Memoria de Corto Plazo o *Short Term Memory* (STM) de V2. Aunque PBSeg realiza la segmentación solo en los objetos de fondo, los resultados en $Y_{zc}(i,j)^{n_z}$ pueden variar entre un cuadro y otro como se muestra en la Figura 5.38. Estas diferencias se deben a variaciones en matiz que se generan en el mismo video, aunque en otras ocasiones pueden ser debido a los ajustes que realiza el módulo V4 que se documenta en la siguiente subsección.

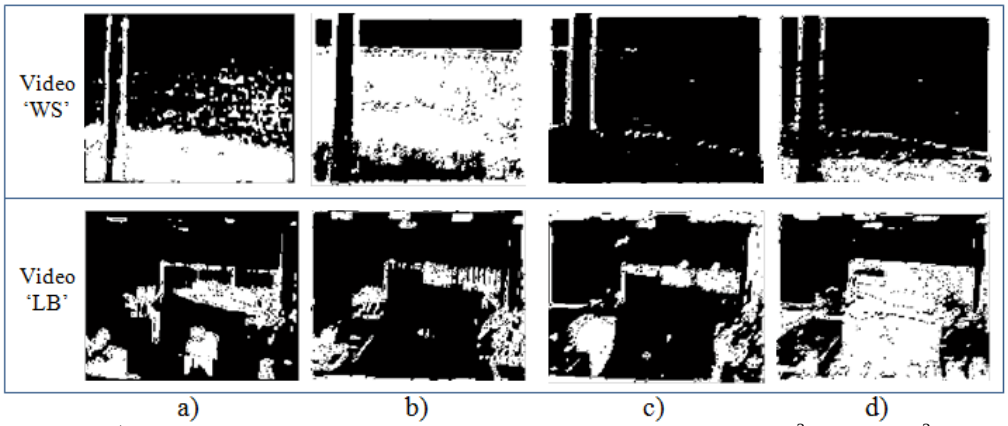


Figura 5.37. $Y_{zc}(i,j)^{tz}$ cuando se estima σ_{zc} con las ecuaciones 5.51 y 5.56. a) $Y_{\varsigma Y}(i,j)^3$. b) $Y_{\varsigma G}(i,j)^3$. c) $Y_{\varsigma B}(i,j)^3$. d) $Y_{\varsigma R}(i,j)^3$.

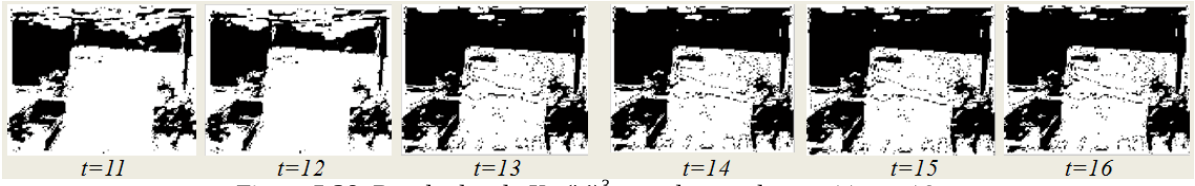


Figura 5.38. Resultados de $Y_{\varsigma R}(i,j)^3$ entre los cuadros *t=11* a *t=16*.

Para reducir errores debido a estas variaciones, PBSeg tiene un submódulo que almacena el resultado de los últimos diez cuadros. Esta información se utiliza como entrada en V4 para que esta genere la segmentación de objetos estáticos a partir de la información de varios cuadros. Este submódulo va acumulando el resultado de la siguiente forma:

$$STM\left(i,j,n_1\right)^{z_c} = Y_{z_c}\left(i,j\right)^3 \tag{5.57}$$

donde n_1 es el índice de la memoria $STM(i,j,n_1)^{z_c}$, y en cada cuadro $n_1=n_1+1$ y se regresa a $n_1=1$ cada que $n_1=N_1$ (tamaño de la memoria $STM(i,j,n_1)^{z_c}$, la cual se inicializa en $N_1=10$). Con base a la ecuación 5.57, se obtiene una memoria que contiene los últimos diez resultados de $Y_{zc}(i,j)^{n_z}$. De esta manera, $STM(i,j,n_1)^{z_c}$ se asemeja a una memoria FIFO como se ve en la Figura 5.39. Esta memoria $STM(i,j,n_1)^{z_c}$ y el agrupamiento perceptual $\Gamma_q(i,j)$ son la salida del módulo V2. Este módulo se utilizará para generar información de memoria de corto plazo.

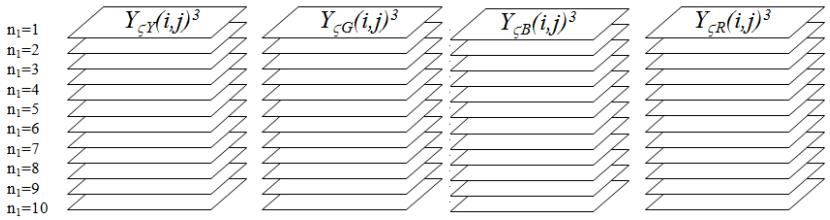


Figura 5.39. Arquitectura de $STM(i,j,n_1)^{z_c}$.

5.2.5 Módulo V4.

Este módulo tiene como objetivo generar la segmentación estática, tal como en el caso del área visual 4 (V4) detecta objetos. De acuerdo a [149], V4 realiza una detección de objetos a partir de color y luminancia, por lo que el módulo V4 de PBSeg utiliza la información derivada de luminancia y de color para obtener la segmentación de objetos dinámicos. Este módulo está compuesto de tres capas:

- *Etapa cortical de entrada*. Procesa los resultados de la $STM(i,j,n_1)^{z_c}$ que contiene información espacio-temporal de los canales $z_c=\{\varsigma_R,\varsigma_G,\varsigma_B,\varsigma_Y\}$ que deberán utilizarse para la segmentación de objetos estáticos.
- *Segmentación de objetos*. Esta etapa realiza la integración de la información de color-textura para la segmentación de objetos estáticos.
- *Ajuste de parámetros con base a agrupamiento perceptual y gradiente de color*. Compara el resultado de la segmentación de color-textura con el resultado del agrupamiento perceptual. Si esta comparación indica coherencia en la segmentación, entonces la actual segmentación de objetos se define como la segmentación estática. Si no, entonces este módulo ajusta parámetros hasta obtener coherencia en el resultado de la segmentación de color-textura. Esta capa tiene un enfoque cognitivo; es decir, se

puede valer de conocimiento generado por resultados pasados del método PBSeg en la misma secuencia de video.

Etapa cortical de entrada. Esta etapa procesa la información de $STM(i,j,n_1)^{z_c}$ para obtener un conjunto de imágenes binarizadas que representen la información contenida en los canales de color de $V_n(x,y)^{z_c,t}$. En el primer paso, el módulo V4 obtiene un promediado de $STM(i,j,n_1)^{z_c}$ dado por:

$$V_4(x,y)^{z_c,t} = \frac{1}{N_1}\sum_{n_1=1}^{N_1} STM\left(i,j,n_1\right)^{z_c} \tag{5.58}$$

Como se puede observar en la Figura 5.40, $V_4(x,y)^{z_c}$ es un conjunto de imágenes que se encuentran en escala de grises.

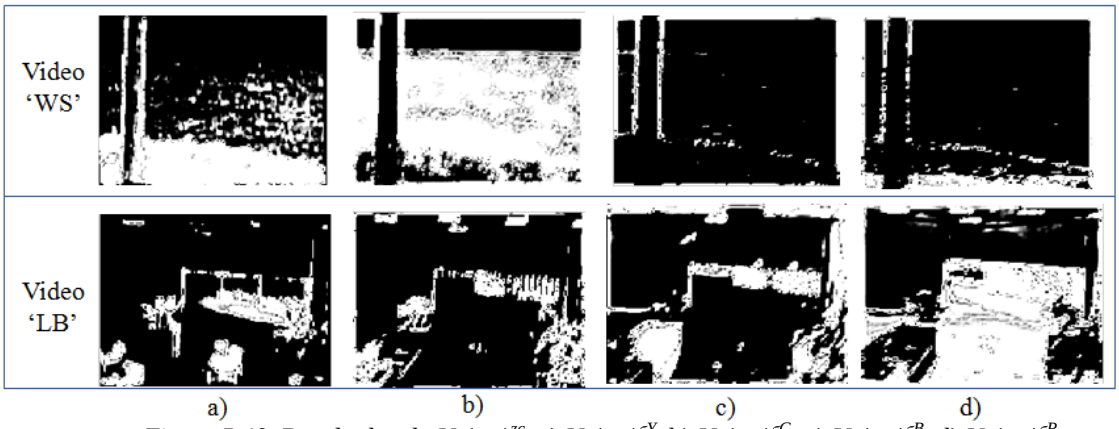


Figura 5.40. Resultados de $V_4(x,y)^{z_c}$. a) $V_4(x,y)^{\varsigma Y}$. b) $V_4(x,y)^{\varsigma G}$. c) $V_4(x,y)^{\varsigma B}$. d) $V_4(x,y)^{\varsigma R}$.

Para obtener imágenes que representen los objetos en los canales de color de $V_n(x,y)^{z_c,t}$, es necesario binarizar el resultado de $V_4(x,y)^{z_c,t}$ con el objetivo de que estos puedan generar regiones que representen cada uno de los objetos en la imagen. Para ello, se recurrió a la NTCNN para representar el módulo V4 con una red neuronal, la cual tiene como utilidad binarizar $V_4(x,y)^{z_c,t}$. Esta capa cortical tiene las mismas condiciones que la NTCNN definida en el capítulo III. Sin embargo, en este caso se va a tener una NTCNN para cada canal de color z_c.

Entrada: Imagen $I(x,y)$ de ceros en todos los canales de color.

Estado inicial: $\chi(i,j,0)^{z_c}=V_{4c}(x,y)^{z_c}$, donde $V_{4c}(x,y)^{z_c}=1-2V_4(x,y)^{z_c}$, donde $x=i$, $y=j$.

En este caso, el parámetro a_2 de la NTCNN es cero, ya que las regiones generadas en $V_4(x,y)^{z_c,t}$ no contienen ruido tan extendido de manera que sea necesario analizar los vecinos N_{25} de cada pixel. Además, si $a_2>0$, entonces se pueden unir regiones que representen objetos distintos. El parámetro a_1 de la NTCNN se dejo constante en $a_1=0.2$ y $z_o=0$, ya que valores $a_1>0.2$ y $z_o<0$ causan que regiones que representen objetos distintos se unan. Sin embargo, conforme disminuya el valor de a_1 y aumente el valor de z_o, existe la posibilidad de dejar más ruido (pixeles aislados con nivel de grises de cero dentro de regiones) en las regiones que representen objetos. Por lo tanto, el modelo cortical NTCNN queda definido de la siguiente manera:

$$\chi_{z_c}(i,j)^{n_c+1} = 2f\left[\chi_{z_c}(i,j)^{n_c}\right] + 0.2^{n_c}\sum_{(i_N,j_N)\in N_8} f\left[\chi_{z_c}(i_N,j_N)^{n_c}\right] \tag{5.59}$$

La salida cuando $y_{zc}(i,j)\rightarrow\infty$ se define de la siguiente forma:

$y_{zc}(i,j)=1$, si $V_{4c}(x,y)^{z_c} > zc+Nac$.

$Y_{zc}(i,j)=0$, si $V_{4c}(x,y)^{z_c} \leq zc+ Nac$.

Donde $Nac = 0.2^{n_c}\sum_{(i_N,j_N)\in N_8} f\left[\chi(i_N,j_N)^{n_c}\right]$.

Finalmente, después de 8 iteraciones (NC=8), la salida de las NTCNN genera un conjunto de imágenes binarizadas en cada cuadro dada por:

$$V_{b4}(x,y)^{z_c,t} = \left[y_{z_c}(i,j)^8 - 1\right]/2 \tag{5.60}$$

En la Figura 5.41 se puede observar el resultado de $V_{b4}(x,y)^{z_c}$. El video '*WS*' tiene un matiz principalmente azul, pero en $V_{b4}(x,y)^{z_c}$ se pueden reconocer las regiones de el mar, la tierra, el zacate, mientras que el poste y el cielo son solo bordeados debido a su baja saturación. El escenario del video '*LB*' contiene objetos que son mapeados en todos los canales de $V_{b4}(x,y)^{z_c,t}$, ya que las plantas se mapean en el canal ς_G, los objetos de madera en en el canal ς_Y, los sillones magenta en el canal ς_B y el piso en el canal ς_R.

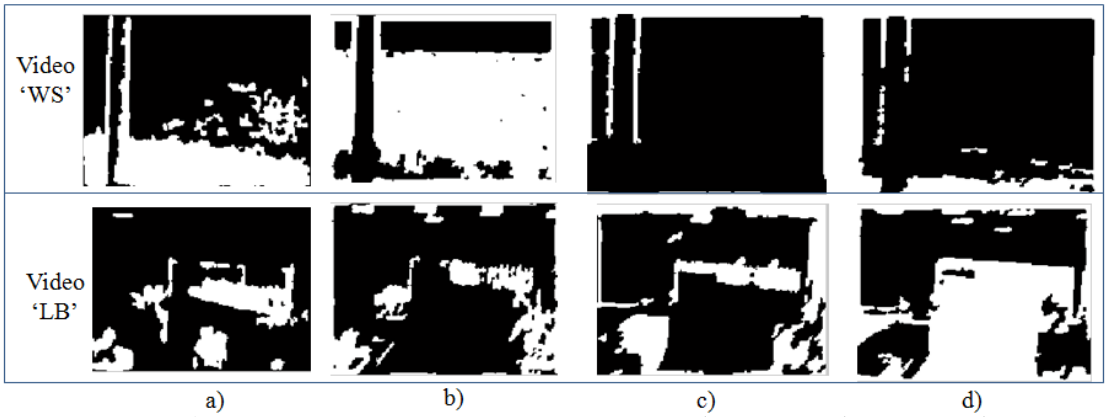


Figura 5.41. $V_{b4}(x,y)^{zc}$ de los videos 'WS' y 'LB' en t=22. a) $V_{b4}(x,y)^{\varsigma_Y}$. b) $V_{b4}(x,y)^{\varsigma_G}$. c) $V_{b4}(x,y)^{\varsigma_B}$. d) $V_{b4}(x,y)^{\varsigma_R}$.

Como se puede ver en las Figuras 5.40 y 5.41, la NTCNN puede eliminar el ruido contenido en las regiones, obteniéndose una representación más coherente de los objetos en el escenario. Además, $V_{b4}(x,y)^{z_c,t}$ realiza una representación de los objetos más estable que $Y_{zc}(i,j)^{n_z}$ ya que representa la información de los objetos estáticos en una ventana de tiempo de tamaño $N_1=10$, eliminándose problemas de ruidos transitorios. Un aspecto interesante es que el resultado en $V_{b4}(x,y)^{z_c,t}$ forma regiones que no solo se definen por color, también por textura. Esto se puede ver en el ejemplo del video '*WS*' donde los objetos de fondo cielo y mar tienen el mismo nivel de matiz y color, pero diferente textura. No obstante, como se puede ver en la Figura 5.41, el mar genera una región en $V_{b4}(x,y)^{\varsigma_G}$ causada por la estimulación de los RFO en $V_n(x,y)^{z_c}$ ya que la distribución de los valores que representan el mar generan diferentes niveles de saturación y valor, mientras que el cielo no se estimuló en los RFO debido a que tiene una saturación homogénea y baja, causando que ningún canal en $V_{b4}(x,y)^{z_c}$ genere una región a partir del cielo. Las razones por las cuales se puede segmentar por textura son, primero, la separación de la información espacial en los RFO, segundo, porque en V2 se estima el tamaño de W_{zc} con el objetivo de que cada canal tendrá su propio mecanismo de generación de pulsos en GSCM con base a las características de los objetos presentes en el escenario. Por lo tanto, las regiones generadas desde $Y_{zc}(i,j)^{n_z}$ están definidas a partir del color del objeto y sus dimensiones. El color es definido mediante la localización de las regiones que representan al objeto en los canales de color, mientras que las dimensiones de las regiones permiten definir el funcionamiento de la GSCM, de manera que los pulsos de salida pueden separarse en función de las características de los objetos.

Segmentación de objetos. Una vez calculado $V_{b4}(x,y)^{z_c,t}$, el siguiente paso es obtener los resultados de la segmentación de objetos. $V_{b4}(x,y)^{z_c,t}$ contiene regiones definidas con base a la información de color y textura de los objetos en una ventana de tiempo de tamaño N_1. Sin embargo, el problema en este conjunto de regiones $V_{b4}(x,y)^{z_c,t}$ es que se puede generar más de una región para representar un objeto. Un ejemplo se puede observar en el video '*LB*', donde varios objetos resaltados con círculos de colores forman regiones que se repiten en varios canales de $V_{b4}(x,y)^{z_c,t}$. En el escenario del video '*LB*', las lámparas se repiten en todos los canales, las plantas en los canales amarillo y verde, la rejilla en los canales verde, azul y rojo, el mueble del centro de mesa en amarillo y rojo. Este problema de repetición de regiones se da en todas las secuencias de video probadas, por lo tanto, antes de generar el cuadro que representa los objetos en la secuencia de video se debe realizar un análisis para encontrar cual región representa más coherentemente un objeto.

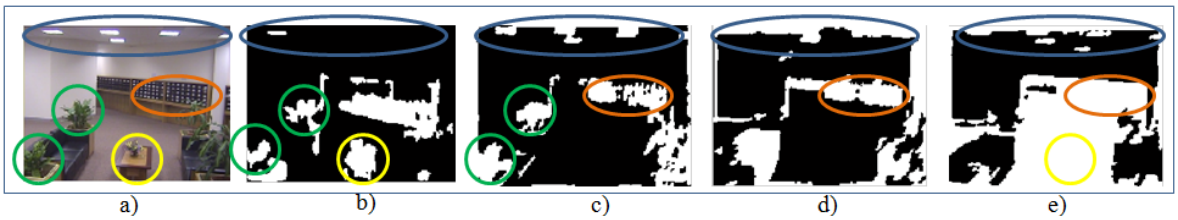


Figura 5.42. Repetición de regiones por objeto en el escenario de una secuencia de video. a) Cuadro *t=22*, Video '*LB*'. b) $V_{b4}(x,y)^{\varsigma_Y}$. c) $V_{b4}(x,y)^{\varsigma_G}$. d) $V_{b4}(x,y)^{\varsigma_B}$. e) $V_{b4}(x,y)^{\varsigma_R}$.

Selección de región por objeto con base a matiz. Para realizar la selección de la región de cada objeto, PBSeg se puede valer de la información de matiz H. Por lo tanto, antes de la integración de regiones se realiza un análisis en cada píxel $p_e(x,y) \epsilon V_{b4}(x,y)^{z_c,t}$ para verificar cuantos canales z_c se encuentran activos en este píxel $p_e(x,y)$ y si se tiene más de un canal activo, analizar con base a matiz H cuál es que se debe mantener activo e inhibir el resto. No obstante, primero se debe analizar si los niveles de matiz H son correspondientes a la información contenida en los campos receptivos de color. Por lo tanto, para reducir la cantidad de canales se realiza el siguiente algoritmo:

1. Encontrar el valor esperado de la diferencia absoluta entre $hI(q)$ y $hC_r(q)$ para verificar la distribución entre los canales de los campos receptivos de color y matiz. Aunque anteriormente se verificó la relación matiz / distribución de campos receptivos con la ecuación 5.45, el propósito de la ecuación 5.61 es medir que tan similares son $hI(q)$ y $hC_r(q)$. Este valor esperado esta dado por:

$$\mu_h = \frac{1}{4}\sum_q \left| hI(q) - hC_r(q) \right| \tag{5.61}$$

En la mayoría de los casos donde existe consistencia entre matiz dado por *hI(q)* y campos receptivos de color $hC_r(q)$, μ_h tiene valores de *(0,0.1)*. De acuerdo a los experimentos realizados, la consistencia entre *hI(q)* y $hC_r(q)$ se pierde en escenarios obscuros; en este caso los niveles de matiz H del espacio HSV, no son muy descriptivos respecto a los objetos de la escena ya que la información de valor es baja y en este caso la información de H y S pierden coherencia como se vio en la sección 4.1 del capítulo IV. Sin embargo, en condiciones de obscuridad los campos receptivos de color tienen mejor representación del color que H, ya que tal como se vio en la Figura 5.19, estos son variantes a H y S pero invariantes a luminancia.

2. Si $\mu_h > 0.1$, entonces se procede al paso 4. Si no, se genera una imagen de matiz agrupado dada por:

$$I_h(x,y) = \begin{cases} \lceil 4H(x,y)/2\pi \rceil / 4 & hCr = hI_{rb} \\ \lceil 4H_3(x,y)/2\pi \rceil / 4 & hC_r = hI_r \end{cases} \tag{5.62}$$

$I_h(x,y)$ contiene la información de matiz agrupada en cuatro clases, las cuales son *{amarillo,verde,azul,rojo}* definidas en cada píxel $p_e(x,y) \epsilon I_h(x,y)$ como $p_e(x,y)=\{0.25,0.5,0.75,1\}$ respectivamente.

3. Definir cuántos canales se encuentran activos en cada píxel $p_e(x,y) \epsilon SV_4(x,y)$ mediante lo siguiente:

$$SV_4(x,y) = \sum_{z_c} V_{b4}(x,y)^{z_c,t} \tag{5.63}$$

Con base a la ecuación 5.63, si un píxel $p_e(x,y) \epsilon SV_4(x,y) > 1$, entonces tiene más de un canal activo. Bajo esta condición, se debe analizar si este pixel se va a mantener así o se va a dejar solo un canal activo en el píxel $p_e(x,y) \epsilon SV_4(x,y)$. Haciendo un análisis de los colores que representan los campos receptivos de color, se puede observar que en los casos donde $p_e(x,y) \epsilon SV_4(x,y) > 2$, este tiene una saturación baja que logra activar los RF de color. Esto sucede por ejemplo en las lámparas y la rejilla del video '*LB*'; en este caso se puede recurrir a la información de H para definir cual RF va a representar dicho pixel. En los casos donde

$p_e(x,y) \epsilon SV_4(x,y)=2$ se pueden tener colores cyan, magenta, verde, amarillo, rojo, café o anaranjado. Los colores cyan, magenta y verde se pueden detectar fácilmente con H. Sin embargo, los colores café, anaranjado, amarillo y rojo no siempre se puede detectar con H ya que estos colores tienen niveles de H muy similares. En la Tabla 5.1 se muestra cómo se mapean los distintos colores en los campos receptivos de color, donde se puede apreciar que los RF ς_R, ς_Y mapean más colores perceptualmente visibles que los RF ς_G, ς_B.

El color rojo tiene un valor más alto en ς_R que en ς_Y, el amarillo tiene un valor más alto en ς_Y que en ς_R. El color anaranjado tiene casi el mismo valor en ς_Y y ς_R. El color café es un anaranjado con baja saturación, por lo tanto tiene valores bajos en ς_Y y ς_R. En consecuencia, si $p_e(x,y) \epsilon SV_4(x,y)>2$ o si $p_e(x,y) \epsilon SV_4(x,y)=2$ y los canales de $V_{b4}(x,y)^{z_c}$ no son ς_R y ς_Y, entonces se determina el RF de color con H. Si $p_e(x,y) \epsilon SV_4(x,y)=2$ y los colores de $p_e(x,y)$ son amarillo o rojo, entonces, se utiliza más pulsos del canal rojo para la segmentación de objetos estáticos.

Tabla 5.1. Posición del color en los campos receptivos.

Color	Canal z_c
Rojo	ς_R o ς_R-ς_Y
Anaranjado	ς_R o ς_Y
Café	ς_R o ς_Y
Amarillo	ς_Y o ς_Y-ς_R
Verde	ς_G o ς_Y-ς_G
Cyan	ς_B o ς_G
Azul	ς_B o ς_B-ς_G
Magenta	ς_R o ς_R-ς_B

En consecuencia, para seleccionar un canal en los pixeles de $V_{b4}(x,y)^{z_c}$ que se pueden determinar con base a H, se realiza lo siguiente: se genera un arreglo de ceros $V_{a4}(x,y)^{z_c}$ y después se aplican las siguientes ecuaciones:

$$YRc = \left(SV_4\left(x_1,y_1\right) > 2 \,|\, \left(SV_4\left(x_1,y_1\right) = 2 \,\&\, \left(V_{b4}\left(x_1,y_1\right)^{\varsigma_G} = 1 \,|\, V_{b4}\left(x_1,y_1\right)^{\varsigma_B} = 1 \right) \right) \right) \quad (5.64)$$

$$V_{a4}\left(x_1,y_1\right)^{z_c} = \begin{cases} V_{a4}\left(x_1,y_1\right)^{\varsigma_Y} = 1 \quad si \quad \left[Ih(x_1,y_1) = 0.25 \,\&\, YRc \right] \\ V_{a4}\left(x_1,y_1\right)^{\varsigma_G} = 1 \quad si \quad \left[Ih(x_1,y_1) = 0.5 \,\&\, YRc \right] \\ V_{a4}\left(x_1,y_1\right)^{\varsigma_B} = 1 \quad si \quad \left[Ih(x_1,y_1) = 0.75 \,\&\, YRc \right] \\ V_{a4}\left(x_1,y_1\right)^{\varsigma_R} = 1 \quad si \quad \left[Ih(x_1,y_1) = 1 \,\&\, YRc \right] \\ V_{b4}\left(x_1,y_1\right)^{z_c} \quad otra \quad forma \end{cases} \quad (5.65)$$

donde (x_1,y_1) es cualquier píxel de $V_{a4}(x,y)^{z_c}$ y $V_{b4}(x,y)^{z_c}$. La ecuación 5.65 activa solo uno de los canales si existen más de dos canales activos en $V_{b4}(x_1,y_1)^{z_c}$ o existen dos canales activos que no son ς_R o ς_Y. En la Figura 5.43 se puede observar el resultado de este análisis para el video 'LB'. Adicionalmente, si existen regiones de rojos y amarillos, entonces es necesario analizar si se va a agregar más información de los pulsos, lo cual es parte del siguiente paso.

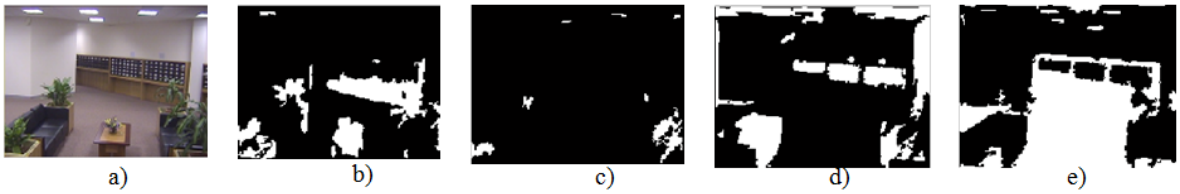


Figura 5.43. $V_{a4}(x,y)^{z_c}$ después de la reducción de color. a) Cuadro '*LB*', *t=22*. b) $V_{a4}(x,y)^{\varsigma_Y}$. c) $V_{a4}(x,y)^{\varsigma_G}$. d) $V_{a4}(x,y)^{\varsigma_B}$. e) $V_{a4}(x,y)^{\varsigma_R}$.

4. Realizar la integración de regiones. En caso que $\mu_h>0.1$, entonces $V_{a4}(x,y)^{z_c}=V_{b4}(x,y)^{z_c}$ (esto sucede generalmente en escenarios obscuros). Si no, se realiza una integración de regiones que inicia basado en $I_h(x,y)$ del punto 2 y $V_{a4}(x,y)^{z_c}$ del punto 3. Luego, realiza la integración de regiones con base a lo siguiente:

Integración de regiones: Una vez obtenido $V_{a4}(x,y)^{z_c}$ en el punto 3, el siguiente paso es integrar las regiones en un cuadro $Is_p(x,y)^t$ que contiene el resultado de la segmentación. Para integrar las regiones de $V_{a4}(x,y)^{z_c}$ en $Is_p(x,y)^t$ se realiza lo siguiente:

$$Is_p(x,y)^t = 0.25*V_{a4}(x,y)^{\varsigma_Y} + 0.5*V_{a4}(x,y)^{\varsigma_G} + 0.75*V_{a4}(x,y)^{\varsigma_B} + V_{a4}(x,y)^{\varsigma_R} \quad (5.66)$$

$Is_p(x,y)^t$ es una primer propuesta de segmentación de objetos estáticos perceptual. Si $V_{a4}(x,y)^{z_c}$ se vio afectada por la ecuación 5.65, entonces $Is_p(x,y)^t$ puede contener hasta seis clases; una clase por cada canal z_c en cada píxel y otra clase definida por la suma de los canales ς_R y ς_Y. Hasta este punto, PBSeg puede generar seis clases para diferentes colores como azul o cyan, verde o cyan, magenta, amarillos, anaranjados y regiones no saturadas. En regiones amarillas, se activa solo $V_{a4}(x,y)^{\varsigma Y}$ en regiones anaranjadas y rojas se suman $V_{a4}(x,y)^{\varsigma R}$ y $V_{a4}(x,y)^{\varsigma Y}$ ya que la ecuación 5.65 no considera ese caso. En consecuencia la ecuación 5.66 puede confundir colores como rojos, y anaranjados. La información que puede indicar si un objeto es rojo esta en $Y_{\varsigma R}(i,j)^2$, ya que si el objeto es amarillo o anaranjado entonces la información en ese pulso del canal ς_R es nula. Si un objeto presente en el escenario es rojo, entonces se presentarán regiones en $Y_{\varsigma R}(i,j)^2$. Esto es debido a que el

nivel de grises de rojo y amarillo en el canal $V_n(x,y)^{\varsigma R}$ es muy diferente como se puede ver en la Figura 5.44, ya que el amarillo genera una respuesta con bajos niveles de grises en ς_R y el rojo genera una respuesta con altos niveles de grises en ς_R. Si la información es amarilla o anaranjada, los niveles de grises iniciarán su actividad en la GSCM a partir del tercer pulso, o la información de $Y_{\varsigma R}(i,j)^2$ e $Y_{\varsigma R}(i,j)^3$ serán iguales. Por tanto, si el valor esperado de $Y_{\varsigma R}(i,j)^2$, $\mu_{2\varsigma R}$, es diferente de cero, o el coeficiente de correlación de *Pearson* entre $Y_{\varsigma R}(i,j)^2$ y $Y_{\varsigma R}(i,j)^3$ es menor uno, se realiza lo siguiente:

$$Is_p(x,y)^t = Is_p(x,y)^t + 0.25 * Y_{\varsigma_R}(i,j)^2 \tag{5.67}$$

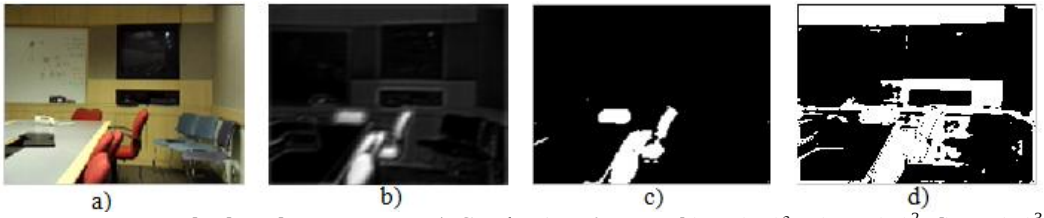


Figura 5.44. Ejemplo de pulsos en $z_c=\varsigma_R$. a) Cuadro '*MO*', *t=22*. b) $V_n(x,y)^{\varsigma R}$. c) $Y_{\varsigma R}(i,j)^2$. d) $Y_{\varsigma R}(i,j)^3$.

Si $V_{a4}(x,y)^{z_c}$ no se analizó con $I_h(x,y)$ entonces $Is_p(x,y)^t$ puede tener hasta 16 clases que se dan de la combinación de la activación de los canales z_c al sumarse los valores de los pixeles en la ecuación 5.66.

Una vez obtenida $Is_p(x,y)^t$, esta se normaliza para que el resultado se mantenga en el intervalo de [0,1]. La cantidad de clases depende de la cantidad de colores presentes en el escenario, de manera que si no existen colores no van a existir clases en $Is_p(x,y)^t$, si existen colores que tienen todos los niveles de matiz, pueden llegar a presentarse hasta 7 clases en condiciones normales o hasta 16 clases en condiciones de obscuridad. En la Figura 5.45 se pueden observar los ejemplos del video '*LB*' y '*WS*' y sus resultados de segmentación en $Is_p(x,y)^t$ y $\Gamma_q(x,y)^t$.

Se puede observar que hasta esta capa del módulo V4 del método PBSeg se puede realizar una segmentación de los objetos de fondo con base a características como forma en $\Gamma_q(x,y)^t$, color y textura $Is_p(x,y)^t$. Sin embargo, hasta este punto del método PBSeg no existe una forma de medir la coherencia entre el resultado de $Is_p(x,y)^t$ y $\Gamma_q(x,y)^t$. Por lo tanto, la siguiente capa de V4 realiza una medición de coherencia para ajustar algunos parámetros del módulo LGN y mejorar los resultados de la segmentación tomando en cuenta la información de color y luminancia como se sugiere en [149].

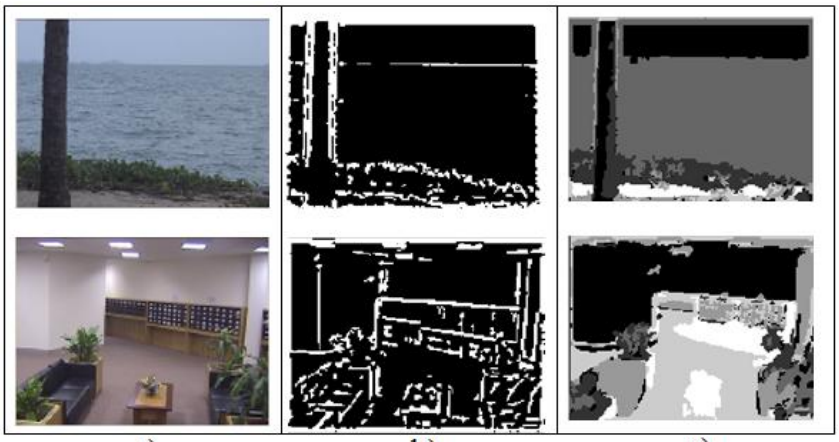


Figura 5.45. Resultados de segmentación de los videos '*LB*' y '*WS*'. a) $I(x,y,z)^t$. b) $\Gamma_q(x,y)^t$ de V2. c). $Is_p(x,y)^t$ de V4.

Ajuste de parámetros con base a agrupamiento perceptual y gradiente de color. Esta capa realiza los ajustes necesarios en el resto de los módulos para asegurar los mejores resultados posibles en la segmentación de objetos en PBSeg; es decir, caracteriza el comportamiento de PBSeg. Adicionalmente, esta capa define el comportamiento de otros módulos caracterizando parámetros como N1, iteraciones en GSCM, el tamaño de los campos receptivos de color y ON/OFF, orientación, los LGN, los parámetros de la RESOMV1 y si es necesario realizar el agrupamiento perceptual.

Para realizar los ajustes, PBSeg mide la coherencia entre el resultado luminancia y color; ya que aunque son características muy distintas, ambas vienen del mismo escenario y en consecuencia, deben estar relacionadas. Idealmente, los contornos definidos en $\Gamma_q(x,y)^t$ deben ser los mismos que los bordes generados por las regiones en $Is_p(x,y)^t$, o dicho de otro modo, las superficies definidas en $Is_p(x,y)^t$ deben ser los mismos que las superficies que se podrían generar en $\Gamma_q(x,y)^t$. Los contornos que representan las formas de los objetos de $\Gamma_q(x,y)^t$ se forman mediante la SCM que se ajusta automáticamente a partir de la versión de [152]. $Is_p(x,y)^t$ se va generando a partir de varios niveles jerárquicos que incluyen diferentes modelos como la GSCM, la NTCNN, la memoria FIFO STM y el análisis de matiz en $I_h(x,y)$. En la primer etapa de esta capa de ajuste por agrupamiento perceptual y gradiente de color se mide la coherencia entre la información de color y luminancia para que PBSeg pueda mejorar los resultados de segmentación con conexiones hacia abajo (*top-down*) en caso que sea necesario. Esta comparación supone que la información de los bordes obtenida por luminancia y contenida en $\Gamma_q(x,y)^t$ esta correctamente modelada. En consecuencia,

PBSeg evalúa si la información generada en $Is_p(x,y)^t$ es consistente con la información de $\Gamma_q(x,y)^t$. Para la comparación, se analizó la relación existente entre los bordes contenidos en $\Gamma_q(x,y)^t$ y los bordes de las regiones en $Is_p(x,y)^t$. Esta comparación es un análisis estadístico de la información en frecuencia de $\Gamma_q(x,y)^t$ y el gradiente $Ig_p(x,y)^t$ de $Is_p(x,y)^t$ dado por:

$$Ig_p(x,y)^t = Is_p(x,y)^t * h_1 \tag{5.68}$$

$$h_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.69}$$

El resultado en $Ig_p(x,y)^t$ se binariza para que todos los pixeles diferentes de cero sean uno. En consecuencia, se genera un cuadro que contiene la información de los bordes formados por las regiones en $Is_p(x,y)^t$ como se muestra en la Figura 5.46, donde los bordes en $Ig_p(x,y)^t$ generados por las regiones de $Is_p(x,y)^t$, tienen similitud con los bordes generados en $\Gamma_q(x,y)^t$ que se muestran en la Figura 5.45b. Sin embargo, en ocasiones los bordes en $Ig_p(x,y)^t$ pueden diferir, ya que tal como se ve en la Figura 5.47, en ocasiones el resultado en $Ig_p(x,y)^t$ difiere por completo de $\Gamma_q(x,y)^t$ debido a errores en la segmentación. En el caso del video 'P2009_L1_8' (Figura 2.47), y en la mayoría de los casos donde se presentaron diferencias significativas entre $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$, los errores son causados porque el valor de Θ inicial de los módulos LGN causa errores en el filtrado del color por las condiciones cromáticas del video.

Figura 5.46 $Ig_p(x,y)^t$ de videos a) '*LB*', *t=22*, b) '*WS*', *t=22*.

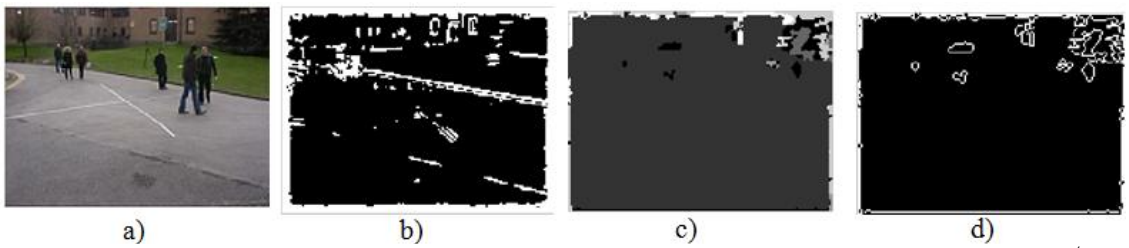


Figura 5.47. Resultados de segmentación de video 'P2009_L1_8' en los primeros cuadros. a) $I(x,y,z)^t$. b) $\Gamma_q(x,y)^t$. c) $Is_p(x,y)^t$. d) $Ig_p(x,y)^t$.

Aunque se puede también modificar el comportamiento de otros parámetros de otros módulos, la versión actual de PBSeg maneja el parámetro Θ de los LGN, y t_a del módulo RESOMV1 para mejorar la coherencia en los resultados de la segmentación. Para verificar si es necesario modificar estos módulos, se hace un análisis para medir la similitud de los bordes de $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$. Para medir esta similitud, primero se realiza una separación en varias imágenes a $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ para analizar el comportamiento de los bordes y su distribución espacial como se muestra en la Figura 5.48, donde se muestran $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ divididos en cuatro secciones que se analizarán por separado para revisar de forma espacial cómo están distribuidos los bordes en $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$. Estas imágenes seccionadas se definen como:

$$V_{f4}(x_a, y_a, w_a), \quad w_a = 1,...,4, \quad x_a = 1,...,X/4, \quad y_a = 1,...,Y/4 \tag{5.70}$$

$$\Gamma_{f4}(x_a, y_a, w_a) \tag{5.71}$$

donde X,Y son los valores máximos de (x,y), ya que $K=XY$, w_a es el índice de sección del cuadro, $V_{f4}(x_a,y_a,w_a)$ es un arreglo que contiene la información de $Ig_p(x,y)^t$ dividido en cuatro secciones y $\Gamma_{f4}(x_a,y_a,w_a)$ es un arreglo que contiene los $\Gamma_q(x,y)^t$ dividido en cuatro secciones. Estas secciones se utilizan para analizar los bordes conociendo su ubicación en $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$. A mayor cantidad de secciones se puede tener una mejor precisión de la distribución espacial de los bordes, no obstante, el tiempo de procesamiento aumenta y no es muy factible tener imágenes con resolución espacial muy baja debido a que se va a utilizar el dominio de la frecuencia para medir la coherencia de los bordes. Para realizar una comparación exitosa se presentan dos problemas: primero, el grosor de los bordes generados en $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ es distinto, segundo, los bordes en $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ no se localizan en los mismos índices de pixeles (es decir, no son punto a punto) aún cuando la información sea coherente entre $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$.

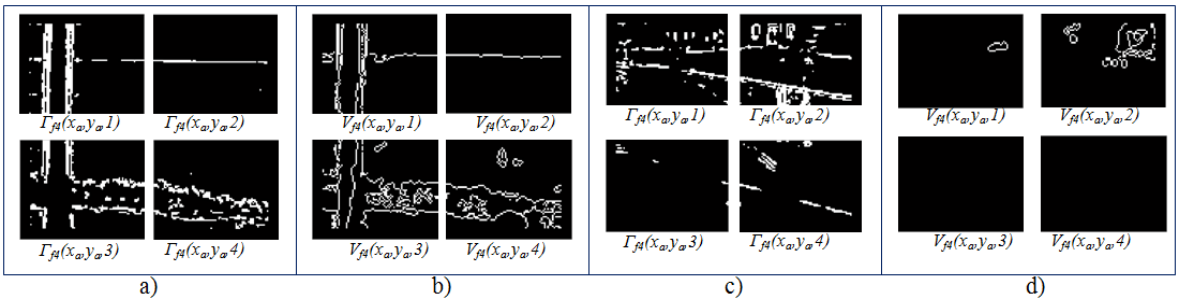


Figura 5.48. Secciones o divisiones de la escena para análisis de bordes. a) $V_{f4}(x_a,y_a,w_a)$ en el video 'WS'. b) $\Gamma_{f4}(x_a,y_a,w_a)$ en el video 'WS'. c) $V_{f4}(x_a,y_a,w_a)$ en el video 'P2009_L1_8'. d) $\Gamma_{f4}(x_a,y_a,w_a)$ en el video 'P2009_L1_8'.

Para analizar los bordes en $V_{f4}(x_a,y_a,w_a)$ y $\Gamma_{f4}(x_a,y_a,w_a)$, se puede recurrir a otros dominios como frecuencia espacial definido por la transformada de Fourier **[**124**]** o al análisis multirresolución defnido por la transformada wavelet [124]. La frecuencia espacial brinda información respecto al contenido de frecuencia de una imagen, pero pierde la información espacial. La transformada wavelet genera información de frecuencia y espacio en diferentes niveles de resolución. En neurociencias, el dominio de la frecuencia espacial se utiliza para simular diferentes fenómenos perceptuales [31], como en el caso de los modelos LISSOM, donde se definen distintos tipos de mapas neurales como orientación (*OR*), color (*CR*), dominancia ocular (*DO*), disparidad (*DY*) y frecuencia espacial (*SF*). Este último se utiliza para simular el comportamiento de ciertas células complejas y para describir el comportamiento de las orientaciones de diferentes neuronas que son sensibles a los gradientes de los mapas neurales de OR y CR **[**143**,**34**,**145**]**. Además, en [154] se menciona que las células complejas de diferentes áreas de la corteza visual procesan la información de una manera muy similar al concepto de frecuencia espacial. El análisis multirresolución se utiliza para simular la descomposición de la información visual en ciertos campos receptivos [31].

Dado que en esta tesis se trabaja con la teoría desarrollada por LISSOM y el concepto de mapas neurales como OR, y CR, y que en este caso, $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ son resultados del análisis de orientaciones, y gradiente de color, se decició realizar la comparación de $V_{f4}(x_a,y_a,w_a)$ y $\Gamma_{f4}(x_a,y_a,w_a)$ en el dominio de la frecuencia espacial, para simular el comportamiento de los mapas SF. Aunque ciertamente los mapas SF no contienen información espacial, esto se compensa con dividir $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ en los arreglos $V_{f4}(x_a,y_a,w_a)$ y $\Gamma_{f4}(x_a,y_a,w_a)$.

En la Figura 5.49, se puede observar la magnitud de la respuesta a la frecuencia $Fv_p(x_a,y_a,w_a)^t$ y $F_{\Gamma q}(x_a,y_a,w_a)^t$ respectivamente de los arreglos $V_{f4}(x_a,y_a,w_a)$ y $\Gamma_{f4}(x_a,y_a,w_a)$ en diferentes escenarios. Si se analiza esta respuesta, se puede obtener una estimación de los patrones formados por los bordes sin importar que estos no sean punto a punto o el grosor de las líneas formadas. En PBSeg cada cuadro está dividido en cuatro secciones para analizar la respuesta a la frecuencia, ya que al aumentar la cantidad de seccionamiento, es menor el espacio analizado y por tanto, los lóbulos de frecuencia más amplios.

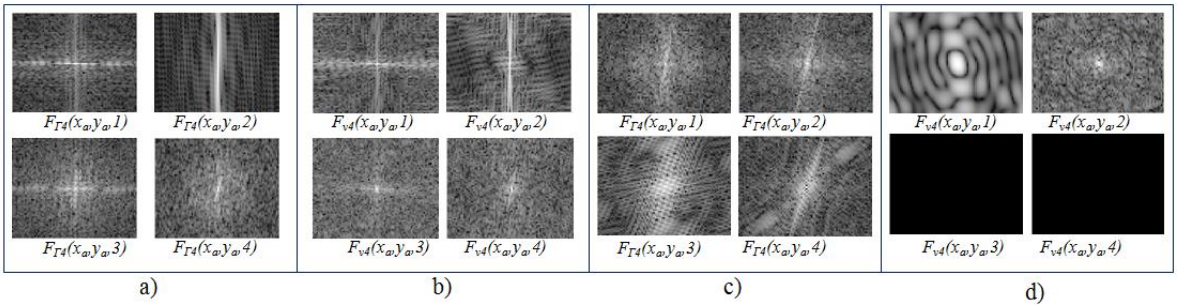


Figura 5.49. Respuesta a la frecuencia de $V_{f4}(x_a,y_a,w_a)$ y $\Gamma_{f4}(x_a,y_a,w_a)$. a) $F_{v4}(x_a,y_a,w_a)$ en el video '*WS*'. b) $F_{\Gamma 4}(x_a,y_a,w_a)$ en el video '*WS*'. c) $F_{v4}(x_a,y_a,w_a)$ en el video '*P2009_L1_8'*. d) $F_{\Gamma 4}(x_a,y_a,w_a)$ en el video '*P2009_L1_8*'.

Cada sección tiene un tamaño de 60x80 y este tamaño es fijo para cualquier resolución; un video de 240x320 tiene 16 secciones de 60x80, un video de 480x640 tiene 64 secciones de 60x80. Es decir, al aumentar la resolución espacial del video, se puede obtener mejor información de la distribución espacial de las frecuencias de los contornos en el escenario.

Basándose en la respuesta a la frecuencia de los bordes, PBSeg obtiene una medición de la coherencia mediante el valor esperado de $F_{vp}(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$, ya que si se forma un patrón muy diferente en frecuencia, el valor esperado va a ser muy distinto, y por lo tanto los niveles de grises también. Esto se puede ejemplificar en la Figura 5.49, donde los resultados de $Fv_p(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$ del video '*WS*' son similares y de acuerdo a los cálculos, también lo es el valor esperado de cada respuesta en $Fv_p(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$ (en la Figura 5.49ª se muestra $Fv_p(x_a,y_a,w_a)$ de 'WS' y en la Figura 5.49b se muestra $F_{\Gamma q}(x_a,y_a,w_a)$ de '*WS*'). No obstante, en el caso del video 'P2009_L1_8' el cálculo del valor esperado difiere por completo en $Fv_p(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$, ya que el patrón formado en los bordes es completamente distinto como se puede observar en la Figura 5.48c y 5.48d. Para medir la similitud del valor esperado se realiza lo siguiente:

$$\mu_f(\omega_a) = \frac{1}{K}\left|\sum_{x_a}\sum_{y_a} Fv_p\left(x_a, y_a, w_a\right) - \sum_{x_a}\sum_{y_a} F_{\Gamma q}\left(x_a, y_a, w_a\right)\right| \tag{5.72}$$

Experimentalmente, se determinó que si en cualquiera de las ω_a secciones $\mu_f(\omega_a)>0.5$ entonces la diferencia en bordes en $Ig_p(x,y)^t$ y $\Gamma_q(x,y)^t$ es debida a problemas en la segmentación de objetos. De esta manera, para PBSeg la segmentación de objetos en $Ig_p(x,y)^t$ es coherente si en todas las secciones $\mu_f(\omega_a)\leq 0.5$, pero, si en una sección ω_a $\mu_f(\omega_a)>0.5$, entonces, la segmentación de objetos en $Ig_p(x,y)^t$ no es coherente, y por lo tanto es necesario reajustar parámetros para mejorar los resultados de segmentación.

Ajuste de parámetros: PBSeg ajusta los parámetros por dos razones: una es que cualquiera de los valores de $\mu_f(\omega_a)$ sea menor a 0.5, la cual se acaba de discutir. La otra razón es que exista un cambio muy drástico en las condiciones del escenario. Este cambio drástico se mide de la misma forma que el capítulo III, mediante cambios de entropía con la ecuación 3.52, donde $t_a=0$ si $| E^{tb} - E^t | > DE$. En caso de existir la necesidad de ajuste de parámetros ya sea por $\mu_f(\omega_a)$ o entropía, PBSeg analiza el estado de los parámetros t_a de RESOMV1 y Θ de los módulos LGN con el objetivo de cambiar la forma en cómo los campos receptivos realizan el filtrado de color y que RESOMV1 logre aprender esos cambios rápidamente la nueva respuesta de los LGN. Es decir, si los LGN cambian Θ en el cuadro $I(x,y,z)^{t_1}$, t_a permite que de $V_m(x,y)^{z_c}$ hasta $V_{a4}(x,y)^{z_c}$ aprendan dichos cambios antes de $I(x,y,z)^{t_1+1}$.

El algoritmo de ajuste funciona de la siguiente manera: si en una imagen de $F_{v4}(x_a,y_a,w_a)$ y $F_{\Gamma 4}(x_a,y_a,w_a)$ $\mu_f(\omega_a)$ es mayor a 0.5 durante los primeros N_1 cuadros, o en algún punto del video, o la diferencia de entropía es mayor a DE, entonces $t_a=0$, para que α y σ_β se reinicien (ecuaciones 3.10 y 3.12) y RESOMV1 aprenda la redistribución de la información en los módulos LGN, la cual se realiza reasignando Θ mediante la siguiente relación:

$$\mu_{v4}^{z_c} = \frac{1}{K}\sum_x\sum_y V_{a4}\left(x,y\right)^{z_c,t} \tag{5.73}$$

$$\Theta_{z_c} = c_{v4}\left(\mu_{v4}^{z_c} - \frac{1}{4}\sum_{z_c}\mu_{v4}^{z_c}\right) \tag{5.74}$$

La ecuación 5.73 indica la cantidad de pixeles que se mantienen activos en cada canal de $V_{a4}(x,y)^{z_c,t}$ y la ecuación 5.74 define un parámetro de filtrado Θ_{zc} para cada campo

receptivo que depende de la respuesta de $V_{a4}(x,y)^{z_c,t}$ y el promedio de cantidad de pixeles activos. Esta diferencia permite equilibrar la información de los campos receptivos, ya que la ecuación 5.73 se aplica para generar una nueva respuesta de campos receptivos de color dada por:

$$RF\Theta_{z_c} = RF_m - \Theta_{z_c} \tag{5.75}$$

$$\varsigma_s(RF\Theta_{z_c}) = \tfrac{1}{2}\left(\left|RF\Theta_{z_c}\right| - \left|RF\Theta_{z_c} - 1\right| + 1\right) \tag{5.76}$$

Luego, se vuelve a repetir todo el proceso en los siguientes N_1 cuadros. Inicialmente, c_{v4} es uno y cada que pasan N_1 cuadros, este va bajando su valor en $c_{v4}=c_{v4}-0.2$ hasta que llegue a 0.2 (como $N_1=10$, este proceso lleva 50 cuadros). El parámetro c_{v4} permite probar con diferentes valores de Θ_{zc} hasta encontrar un nivel de filtrado que permita tener un resultado lo más coherente posible.

Este proceso de hacer $t_a=0$ y calcular la respuesta de los LGN en función de la ecuaciones 5.73 a 5.76 cada N_1 cuadros se repite hasta que $\mu_f(\omega_a) \leq 0.5$ en todas las imágenes de $F_{v4}(x_a,y_a,w_a)$ y $F_{\Gamma 4}(x_a,y_a,w_a)$ o hasta que $c_{v4} = 0.2$. Si después que $c_{v4} = 0.2$ no se puede tener un índice de coherencia, entonces se busca el valor de c_{v4} que generan filtrados de colro cuyos resultados de segmentación tienen los valores promediode $\mu_f(\omega)$ más cercanos a 0.5. Con estas ecuaciones, se puede observar un mecanismo similar a los *top-down* de modelos cognitivos como en **[**81**]**, ya que el resultado del módulo V4 de $V_{a4}(x,y)^{z_c,t}$ influye directamente en el comportamiento de los módulos LGN y V1.

Otro aspecto interesante de esta capa es que genera un enfoque cognitivo al cambiar las condiciones del escenario; cuando hay cambios drásticos derivados de cambios de iluminación que son detectados por los cambios de entropía y el escenario se oscurece PBSeg guarda el último resultado de $\Gamma_q(x,y)^t$ en una imagen $\Gamma_c(x,y)^t$ ($\Gamma_c(x,y)^t = \Gamma_q(x,y)^t$), luego reinicia en ceros a $\Gamma_q(x,y)^t$ y obtiene la correlación de los valores nuevos de $\Gamma_c(x,y)^t$ y $\Gamma_q(x,y)^t$ para verificar que los resultados de segmentación nuevos sean similares a los resultados anteriores, ya que un cambio de iluminación no implica un cambio en los objetos de fondo. En la Figura 5.50 se pueden observar tres ejemplos de segmentación por PBSeg; en el caso de los videos ‘LS’ y ‘P2009_L1_8’ fue necesario que la capa de V4 de ajuste de parámetros cambiara los valores de Θ_{zc} para mejorar los resultados de la segmentación. El caso de ‘BT’ muestra que PBSeg puede segmentar tanto por color como por textura, ya que

el agua que forma un Fondo dinámico tiene mucha variabilidad en los valores de los pixeles, pero PBSeg logra generar una región que representa coherentemente el agua. Además, en el video 'BT', los árboles del fondo y el jardín que tienen mismo color, mismo brillo pero diferente textura también son representados coherentemente con las regiones resultantes.

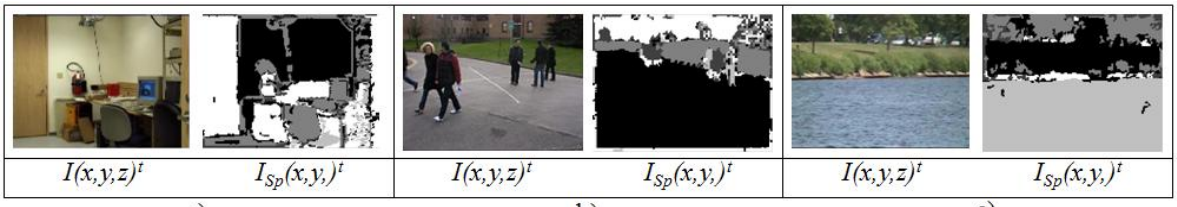


Figura 5.50. Resultados de segmentación con PBSeg. a) Video '*LS*'. b) Video '*P2009_L1_8*'. c) Video '*BT*'.

5.3. Estudio comparativo de segmentación de objetos estáticos

En la literatura aun no se ha reportado un método de segmentación de objetos estáticos neuroinspirados como PBSeg. Incluso como se mencionó en el capítulo IV, en la literatura analizada no existen métodos para segmentación de objetos estáticos. Por lo tanto, el enfoque de PBSeg es una primera aportación que tiene este método, ya que un conjunto de RNA que forman un método inspirado en el funcionamiento de la corteza visual puede segmentar de forma coherente objetos de escenarios reales en secuencias de video.

No obstante, la aportación de PBSeg no termina en este punto, ya que aun no se analiza el resultado de segmentación de este método. Por lo tanto, en esta sección se estudiará la contribución de PBSeg respecto a los resultados de segmentación. Para esto, se realizó un estudio comparativo de los resultados entre PBSeg y otros métodos. Adicionalmente, en esta misma sección se retomará el método KCNN para analizar su contribución como método de segmentación para análisis de video, ya que un objetivo de incluir KCNN en esta tesis es comparar el resultado de un enfoque clásico con un enfoque neuroinspirado para la segmentación de ojetos estáticos. Por lo tanto, los métodos para realizar el estudio comparativo de PBSeg son KCNN de capítulo IV, *Subtracting Clustering* (SC) [155], *Graph-Based Image Segmentation* (GBIS) [156] y *Fractional Order Darwanian Particle Swarm Optimization* (FO DPSO) [157]. Los métodos SC y GBIS son dos métodos ampliamente citados y utilizados en la literatura relacionada a segmentación

de imágenes. FO DPSO es un método que se basa en *Particle Swarm Optimization* (PSO), el cual es ampliamente utilizado para segmentación de imágenes **[**158**]**.

SC. Este método fue seleccionado para reemplazar el *kmeans* y la T4CNN en KCNN ya que el objetivo de SC es encontrar el número de grupos y sus respectivos centros **[**155**]**.

GBIS. Es un método semiautomático que divide una imagen en un conjunto de regiones tomando como base los contornos presentes en la imagen, al igual que los procesos perceptuales [156]. GBIS fue seleccionado no solo porque es muy popular en la literatura, también por su similitud con PBSeg en su filosofía para encontrar regiones. Los parámetros que se deben ajustar en este método son *sigma*, *k* y *min* definidos en **[**156**]**.

FO DPSO. Basado en PSO, este método realiza una segmentación de color. Se trata de un método bioinspirado que es básicamente una versión evolutiva del PSO que ajusta sus cálculos mediante el concepto de diferencia fraccional [157]. Es muy utilizado en segmentación de imágenes para aplicaciones médicas **[**159**]**.

La entrada para SC, GBIS y FO DPSO es el modelado de fondo que se realiza en KCNN. El modelado de fondo de PBSeg no se puede utilizar para estos métodos porque PBSeg utiliza su propio espacio de color basado en los LGN el cual contiene más de tres canales. No se pudieron utilizar otros métodos relacionados a segmentación en secuencias de videos ya que en la literatura no se reportan métodos sobre segmentación de objetos de fondo.

Los videos utilizados en los experimentos fueron los mismos que se utilizaron en el capítulo III, los cuales tienen *ground truths* disponibles pero solo para segmentación de movimiento. No existen videos reportados en la literatura que tengan *ground truths* que analicen los resultados de segmentar objetos estáticos. Dado que los métodos con los que se va a realizar la comparación son para segmentar imágenes, una opción que se consideró en un principio fue utilizar bases de datos que sirven para estudiar el desempeño de los métodos de segmentación de imágenes. Sin embargo, esta opción se desechó ya que una imagen es invariante en el tiempo y por lo tanto el escenario contenido en la imagen no cambia. En consecuencia, con imágenes digitales no se puede medir la coherencia de la segmentación en el tiempo y no se puede caracterizar totalmente la contribución de PBSeg y KCNN ya que estos métodos segmentan el fondo en videos que por definición son

variantes en el tiempo y por lo tanto el escenario puede cambiar. Sin embargo, la contribución de PBSeg y KCNN resulta evidente en procesamiento de video cuando se realiza una comparación cualitativa de los resultados obtenidos entre estos y los métodos para segmentar imágenes fijas, ya que estos últimos tienen problemas evidentes respecto a la coherencia en el tiempo y en la correcta detección de los objetos presentes en el escenario.

Para mostrar los resultados de segmentación se eligieron los videos de '*MO*', '*P2001_1*', '*WS*', '*V1*', '*LB*' y '*TD*' ya que contienen escenarios variantes en el tiempo con situaciones muy distintas que pueden afectar el resultado de la segmentación. Se realizaron más experimentos, pero por motivos de espacio solo se documentan los videos mencionados.

Video '*MO*': Este video de 1745 cuadros contiene un escenario en interior con una pared amarilla, un pizarrón y mesas blancos, un tapiz en la pared color crema, un piso, sillas rojas y azules. A lo largo de la secuencia de video algunos objetos de fondo van cambiando de posición.

El resultado de KCNN muestra coherencia en segmentar el pizarrón, la mesa, objetos en la mesa, sillas y la pantalla, pero tiene un poco de error en la detección del tapiz y el piso. PBSeg tuvo éxito en segmentar la pantalla, la pared, el piso, el tapiz, las sillas rojas, la mesa y los objetos en ella, pero tuvo errores en la detección del pizarrón y las sillas azules. Además PBSeg genera un poco de ruido en los resultados. Ambos, PBSeg y KCNN logran tener coherencia en la segmentación en el tiempo ya que las regiones logran representar en cada cuadro los objetos sin tener cambios en los valores de las regiones, lo cual se puede apreciar en la Figura 5.51 y 5.52 donde se ve que los niveles de grises de $I_{Se}(x,y)^t$ y $I_{Sp}(x,y)^t$ no cambiaron a pesar que existe una diferencia de 1715 cuadros entre el ejemplo de la Figura 5.51 y la Figura 5.52. El método SC tiene buenos resultados con la pantalla, la mesa y algunas sillas y no tiene coherencia en el tiempo. El método FO DPSO tiene resultados coherentes al color pero no a los objetos ya que los objetos son sobresegmentados, es decir, son representados por múltiples regiones. No obstante, FO DPSO mostró coherencia en el tiempo ya que las características cromáticas del escenario no cambian. GBIS no mostró coherencia en segmentar los objetos ni en el escenario ni en el tiempo y aunque genera

resultados interesantes al representar diversas características del escenario, las regiones resultantes no representan los objetos presentes en el escenario. Para GBIS, el mejor resultado obtenido (el cual se muestra en las Figuras 5.51e y 5.52e) *sigma=0.5, ka=500, min=50*.

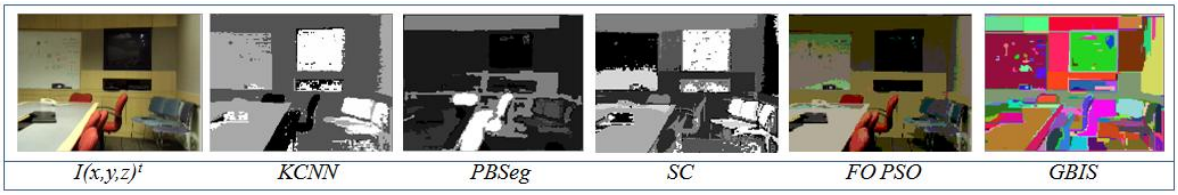


Figura 5.51. Resultados de segmentación de objetos estáticos en el video '*MO*', *t=30*.

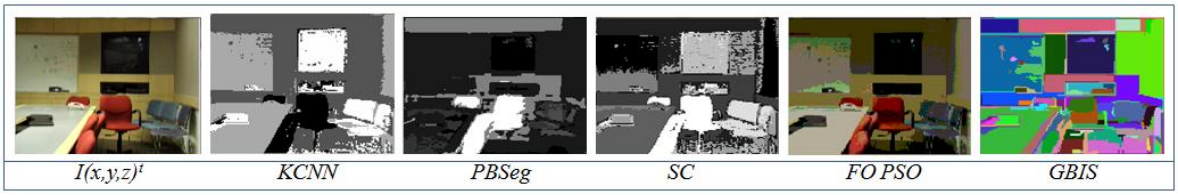


Figura 5.52. Resultados de segmentación de objetos estáticos en el video '*MO*', *t=1745*.

Video '*P2001_1*': Este video de 2687 cuadros contiene un escenario en exterior con varios objetos dinámicos como personas y vehículos. Una característica interesante de este video es que contiene niveles de color distribuidos por toda la banda de matiz.

En este video, KCNN, PBSeg y SC tienen resultados muy similares ya que logran generar buenos resultados en los jardines, la calle, el cielo y el frente de los edificios. Además tienen coherencia en el tiempo como se puede ver en las Figuras 5.53 y 5.54. El caso de FO DPSO también genera resultados coherentes en el tiempo ya que las características cromáticas del escenario no cambian. Aunque FO DPSO sigue coherentemente el color, este no genera resultados coherentes a la representación de los objetos tal como sucede en '*MO*'. GBIS tiene malos resultados en algunos cuadros como se puede observar en la Figura 5.53, pero en otros cuadros logra tener resultados coherentes como en la Figura 5.54. Además, como se puede observar, GBIS no tiene coherencia en el tiempo. Para GBIS, el mejor resultado obtenido (el cual se muestra en las Figuras 5.53e y 5.54e) *sigma=0.5, ka=500, min=50*.

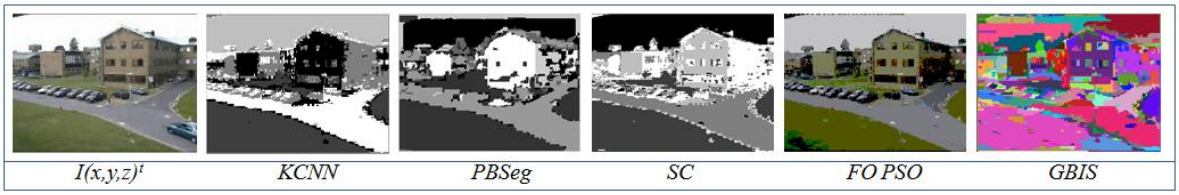


Figura 5.53. Resultados de segmentación de objetos estáticos en el video '*P2001_1*', *t=480*.

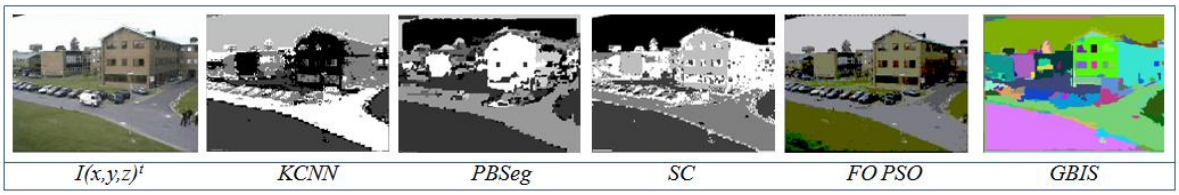


Figura 5.54. Resultados de segmentación de objetos estáticos en el video '*P2001_1*', *t=800*.

Video '*WS*'. Este video de 633 cuadros contiene un escenario en exterior que presenta mar como fondo dinámico, además presenta un objeto dinámico (persona que atraviesa el escenario), arena, vegetación y un tronco. El escenario tiene colores azules principalmente.

En este caso, KCNN tiene resultados satisfactorios en la segmentación del mar, el cielo y el suelo, pero tiene problemas en separar el tronco y el zacate. Con este video se muestra que KCNN logra separar regiones no solo por color, también por textura ya que el cielo y el mar tienen el mismo matiz y color, pero cambia la textura. PBSeg tiene buenos resultados segmentando el mar, el cielo y la vegetación, pero tiene problemas para segmentar el tronco como KCNN. Sin embargo, como se muestra en la Figura 5.56 Pbseg genera en ocasiones un poco de ruido en el resultados de algunos cuadros. SC tiene un desempeño también aceptable, ya que logra segmentar coherentemente el cielo y el mar, pero el suelo, la vegetación y el troco se agrupan en una sola región. Como se puede observar en la Figura 5.55 y 5.56, KCNN, PBSeg y SC generan resultados coherentes en el tiempo. FO DPSO genera buenos resultados en la segmentación del cielo, pero en el mar genera mucho ruido por las diferencias en el reflejo de luminancia presentes en el mar. Este método no tiene coherencia en el tiempo como se puede ver en las diferencias de color del mar y el suelo en *t=20* y en *t=510*. Al igual que KCNN, la vegetación y el tronco se confunden en FO DPSO. GBIS presenta ruido en la región que debe representar el mar y tiene múltiples regiones que representan cada objeto, además las regiones obtenidas por GBIS no son consistentes con el paso de los cuadros (no es coherente ne el tiempo). Para GBIS, el mejor resultado obtenido es con *sigma=0.5*, *ka=1000*, *min=100*.

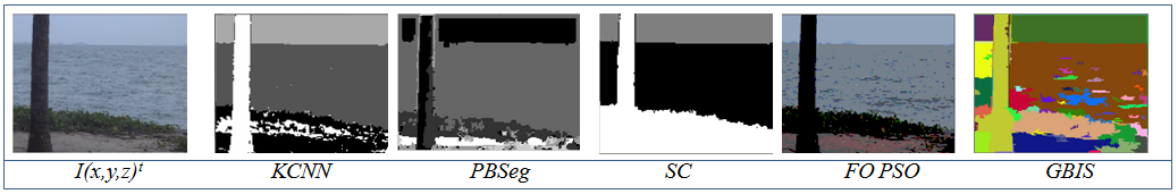


Figura 5.55. Resultados de segmentación de objetos estáticos en el video '*WS*', *t=20*.

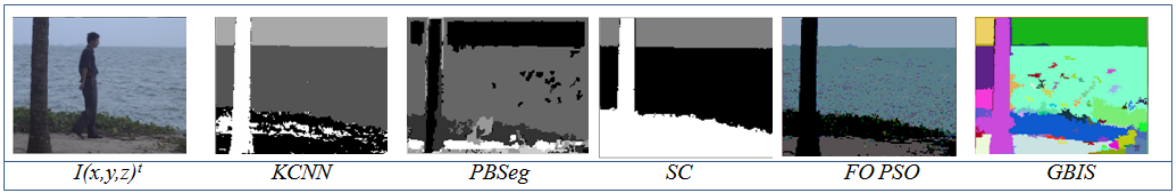


Figura 5.56. Resultados de segmentación de objetos estáticos en el video '*WS*', *t=510*.

Video '*Vi1*': Este video de 252 cuadros tiene un escenario en interior de un pasillo donde las paredes, el techo y el piso tienen los mismos colores. Durante el video se genera un ruido que se puede apreciar como un tinte verde en los niveles de matiz.

KCNN pudo segmentar de manera coherente las puertas, ventanas techos y luces, pero, debido a sombras y problemas de iluminación se generaron errores de segmentación en las paredes. El ruido causado por el tinte verde causa que las regiones que representan las paredes, techo y piso varíen un poco como se ve en la Figura 5.57 y 5.58. El caso de PBSeg fue distinto, ya que debido a que las paredes, el piso y el techo tienen los mismos colores, sombras y reflejos forman una sola región que contiene un poco de ruido. Las ventanas fueron detectadas con ruido y las puertas fueron detectadas correctamente, pero con el problema de que las tres puertas del fondo son representadas por una región. Sin embargo, PBSeg no tiene problemas de variaciones causadas por el ruido del tinte verde como se ve en las Figuras 5.57 y 5.58. SC tiene los mismos problemas de iluminación y sombras en las paredes, pisos y techo que KCNN. Los resultados de KCNN, PBSeg y SC tienen coherencia en el tiempo. FO DPSO tuvo coherencia en seguir los colores, sin embargo, las regiones resultantes no representan ninguno de los objetos en el escenario, y el ruido causó una región verde no deseada en varios cuadros como se ve en la Figura 5.58. GBIS obtiene regiones que representan diferentes patrones en cada cuadro, pero no los objetos presentes en el escenario y no tiene coherencia en segmentar los objetos al pasar los cuadros. Para GBIS, el mejor resultado obtenido *sigma=0.5, ka=500, min=50*.

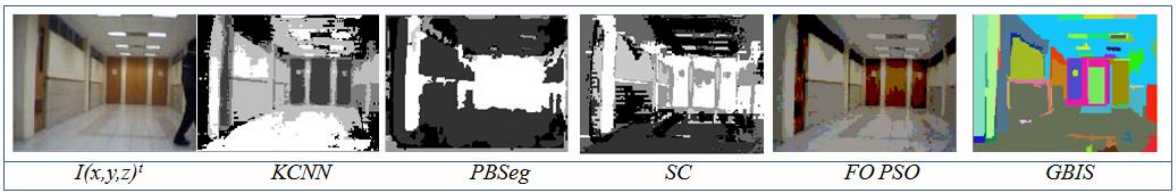


Figura 5.57. Resultados de segmentación de objetos estáticos en el video '*Vi1*', *t=70*.

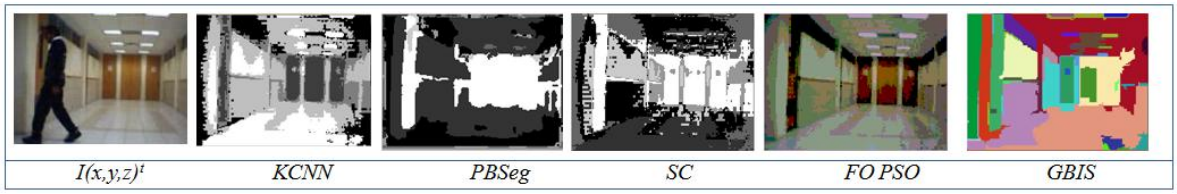


Figura 5.58. Resultados de segmentación de objetos estáticos en el video '*Vi1*', *t=115*.

Video '*LB*': Este escenario de 1546 cuadros contiene un escenario en interiores donde una sala tiene sillones, plantas, mesas, luces, un pilar, una repisa café, una pared blanca y un techo blanco. El video presenta dos cambios de iluminación repentinos. En la Figura 5.59 se puede observar el escenario antes de un cambio luz-obscuridad y en la Figura 5.60 se observa el escenario después del cambio de iluminación.

KCNN logra representar de forma coherente los sillones, las plantas, mesas, el piso, el pilar, la repisa y las luces. No obstante, presenta problemas de sombras y reflejos. Para este video solo KCNN logró tener coherencia en el tiempo. PBSeg logra segmentar de manera coherente las mesas, las plantas, la repisa, el pilar y las paredes. Se puede apreciar en los resultados de PBSeg en *t=350* que en las paredes no se presentan problemas de sombras y reflejos, aunque sí presenta resultados con ruido. Sin embargo, en condiciones de obscuridad donde *t=750*, se generaron errores de segmentación al representar la repisa el piso y la pared en una sola región. Esto sucede porque al obscurecerse el escenario no solo cambia la luminancia, también aumenta la saturación en el RF color rojo, haciendo que los LGN también detecte la pared con color rojizo. Dado que en el video '*LB*' los objetos nunca cambian de color, este cambio de sauración no se genera en el escenario realmente; se debe a la respuesta del sensor de la cámara cuando cambia la iluminación. Por lo tanto, el error de PBSeg se debe más a una cuestión de adquisición del video que a errores en su funcionamiento. SC genera resultados coherentes solo en los sillones, en el resto de las regiones la segmentación no es coherente. FO DPSO genera resultados muy distintos en ambos cuadros ya que cambiaron por completo las condiciones del escenario. Además, este

método también tiene problemas con los reflejos debido a que casi no puede representar los objetos en una región. GBIS no tiene coherencia en el tiempo ya que las regiones generadas cambian en cada cuadro, además las regiones resultantes no representan los objetos contenidos en la escena. Para GBIS, el mejor resultado obtenido (el cual se muestra en las Figuras 5.59e y 5.60e) *sigma=0.5, ka=1000, min=100*.

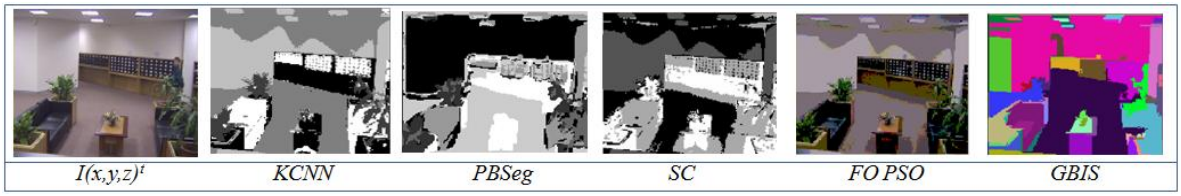


Figura 5.59. Resultados de segmentación de objetos estáticos en el video '*LB*', *t=350*.

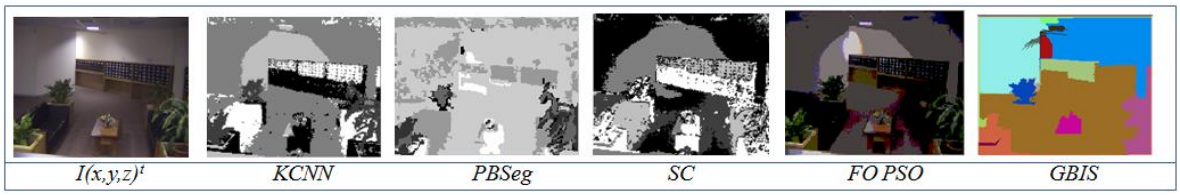


Figura 5.60. Resultados de segmentación de objetos estáticos en el video '*LB*', *t=750*.

Video '*TD*': Este video contiene un cambio de iluminación gradual donde un escenario inicialmente obscuro se va aluzando, simulando un amanecer. Dicha transición va del cuadro 1 hasta el cuadro 1845. El video contiene una pared blanca, dos sillones, una mesa y algunos cuadros en la pared. Este video es útil para mostrar qué sucede con los métodos de segmentación en escenarios obscuros, por lo tanto, para visualizar los resultados se eligieron los cuadros *t=200* (Figura 5.61) y *t=700* (Figura 5.62) en los cuales el escenario aun es muy obscuro.

Respecto a los resultados, en *t=200* KCNN pudo detectar correctamente las paredes, un sillón y la mesa, pero generó mucho ruido en los resultados. En *t=500* KCNN pudo segmentar de manera más coherente los sillones y las paredes. PBSeg logró segmentar un sillón, la mesa y parte de las paredes en *t=200*. En *t=500*, PBSeg detecta correctamente las paredes, los sillones y la mesa. En ambos casos, las regiones resultantes en PBSeg son muy coherentes y con muy poco ruido en comparación con el resto de los métodos. SC segmenta de manera aceptable las paredes y un sillón en *t=200*, mientras que en *t=500* también segmenta adecuadamente la mesa. No obstante, SC genera mucho ruido. El método de FO DPSO genera malos resultados en ambos casos debido a que el escenario es muy obscuro y

los colores no están bien definidos. GBIS tiene malos resultados en *t=200* porque en ese cuadro casi no se aprecian los bordes, pero en *t=500* GBIS tiene resultados aceptables de segmentación. Para GBIS, el mejor resultado obtenido *sigma=0.5*, *ka=500*, *min=50*. Durante los cambios de iluminación gradual, KCNN, PBSeg y SC tuvieron una coherencia regular en el tiempo, mientras que FO DPSO y GBIS no tuvieron coherencia en el tiempo.

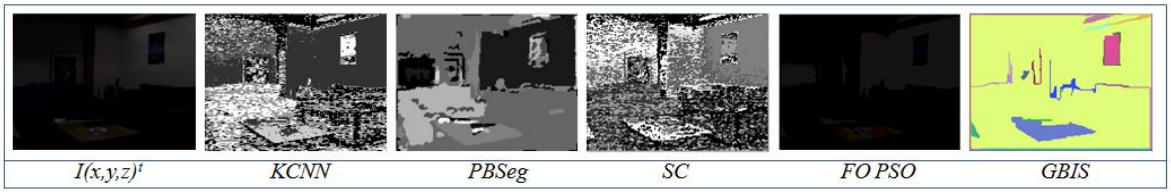


Figura 5.61. Resultados de segmentación de objetos estáticos en el video '*TD*', *t=200*.

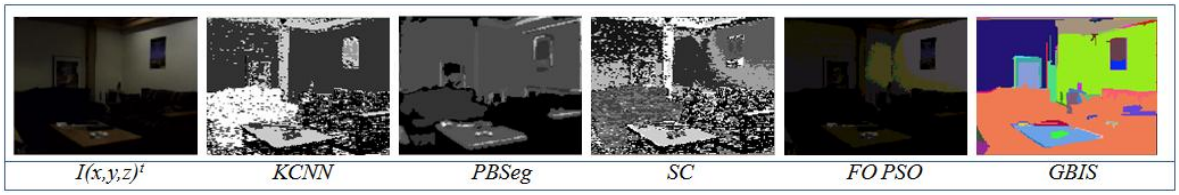


Figura 5.62. Resultados de segmentación de objetos estáticos en el video '*TD*', *t=500*.

Interpretación de los resultados: Con base a los experimentos mostrados y otros realizados, se puede asumir lo siguiente:

SC tiene resultados aceptables respecto a coherencia en representar las regiones como en el tiempo. No obstante, por su definición no es muy confiable para coherencia en el tiempo, ya que en cada cuadro SC puede generar un número de centros distintos y por lo tanto regiones resultantes diferentes. Este método no es recomendable ya que su tiempo de procesamiento para obtener los centros es de 16 segundos en MATLAB® con videos de 120x160.

Los métodos que segmentan color en imágenes como FO DPSO no necesariamente segmentan los objetos presentes en un escenario, ya que en un escenario real los objetos no siempre son de un color y se pueden generar múltiples regiones para representar un objeto. Estos métodos tienen coherencia en el tiempo si el escenario no cambia sus condiciones cromáticas. El método de FO DPSO tiene un tiempo de procesamiento de aproximadamente 9 segundos por cuadro en MATLAB® con videos de 120x160. Este tiempo hace de este y otros métodos de segmentación por color en imágenes poco plausibles para análisis de video.

GBIS puede generar diferentes regiones que generalmente corresponden a múltiples características de los objetos. Esto se puede apreciar en todos los resultados desde la Figura 5.51 hasta la Figura 5.62 ya que las regiones formadas por GBIS tienen información parcial de los objetos, y cada objeto puede ser representado por diferentes regiones. Por lo tanto, para que GBIS logre generar resultados exitosos, es necesario que el método de alguna manera obtuviera información de cómo generar los objetos para poder seguirlos en el tiempo ya que GBIS no tiene coherencia en el tiempo porque en cada cuadro genera resultados muy distintos. El tiempo de procesamiento de GBIS en una implementación de este método en C++ está entre 0.3 a 0.5segundos por cuadro.

El método KCNN demostró en diferentes pruebas ser muy exitoso para segmentar objetos en secuencias de video por diversos motivos. El primero es la segmentación de objetos por cuadro, ya que como se puede ver en las distintas comparaciones, KCNN genera una representación más coherente a los objetos que el resto de los métodos. Segundo, de entre todos los métodos, KCNN es el que muestra mayor coherencia en el tiempo. Tercero, el tiempo de procesamiento genera velocidades de 10 fps, siendo KCNN uno de los métodos de segmentación más rapidos de la literatura. El éxito de KCNN para la correcta detección de objetos radica en el análisis de color (H y S) y luminancia (asociada a V) para encontrar la cantidad de clases en que se puede dividir cada cuadro y las regiones que representan los objetos. El éxito en la coherencia en el tiempo radica primero en la variación de *kmeans* realizada en [**133**] para obtener como centros iniciales los centros resultantes del cuadro anterior; y en segundo lugar, las modificaciones realizadas al *kmeans* en este trabajo relacionadas al análisis de la cantidad de clases para determinar si *kmeans* utiliza como centros iniciales los del cuadro anterior, los de una memoria de centros o que sean aleatorios, así como el ordenamiento de centros y de las regiones resultantes. El tiempo de procesamiento se logra debido a que el cálculo de clases y de centros se realiza cada 60 cuadros, el resto de los cuadros la segmentación se realiza solo con la distancia euclidiana entre los centros y el valor de cada pixel. No obstante, una desventaja de KCNN es su sensibilidad a sombras y reflejos, lo cual causó en ‘LB’, ‘TD’, ‘V1’ y ‘Movil 1’ pérdidas de coherencia en la segmentación.

El método PBSeg tuvo también buenos resultados en segmentar objetos estáticos en secuencias de video. Respecto a la coherencia en representar coherentemente los objetos,

PBSeg tuvo buenos resultados pero con un poco más de ruido que KCNN. Adicionalmente, PBSeg demostró coherencia en el tiempo en diferentes problemas, excepto en cambios de iluminación repentino. El tiempo de procesamiento de PBSeg es aproximadamente de 13 fps. La respuesta de PBSeg en escenarios con bajos niveles de luminancia o son sombras y reflejos es mejor que en el resto de los métodos.

Sin embargo, la contribución del método PBSeg no se limita solo a la segmentación de objetos estáticos. Dado que PBSeg es un método que parte del análisis de diversos modelos neurocomputacionales y diversas teorías perceptuales, este puede encontrar e interpretar diferentes tipos de informaciones contenidas en un escenario. En consecuencia, PBSeg se puede utilizar para segmentar objetos estáticos, contornos, o incluso puede servir como un modelo neurocomputacional. En la siguiente sección se aboraran cada una de las contribuciones de este método de análisis del escenario.

5.4. Conclusiones del método PBSeg

De acuerdo al estudio comparativo, se puede concluir que PBSeg tiene mejor desempeño para segmentar objetos estáticos en secuencias de video que los métodos GBIS, SC y FO DPSO. No obstante, KCNN muestra mejor desempeño en segmentación de objetos estáticos que PBSeg en tiempo de procesamiento y en mostrar coherencia en el tiempo cuando suceden cambios de iluminación. Esto no quiere decir que KCNN sea un mejor método para analizar objetos en secuencias de video que PBSeg, debido a que ambos métodos se plantean a partir de filosofías muy distintas. KCNN es un método que se diseñó con el único objetivo de ser un método que segmenta objetos estáticos en secuencias de video, mientras que PBSeg es un método neuroinspirado que analiza la información de un escenario adquirido desde una secuencia de video y se demuestra su funcionamiento con una aplicación a segmentación de objetos estáticos en secuencias de video.

PBSeg no se limita solo a generar información para segmentar objetos estáticos, ya que adicionalmente, puede encontrar información muy completa de lo que sucede en el escenario. En el módulo más bajo de los LGN, PBSeg descompone la información de cada cuadro en color y luminancia, de manera que la información de color de $zc=\{\varsigma_Y,\varsigma_G,\varsigma_B,\varsigma_R\}$ es variante a saturación y matiz, pero invariante a cualquier cambio de luminancia y valor. Estas características de los RF de color, permiten a PBSeg tener información de los objetos

completamente separada de los niveles de luminancia del cuadro, de manera que en módulos posteriores, se puede generar un resultado robusto a sombras, reflejos y a condiciones de obscuridad. Esta es una primera aportación de PBSeg, el poder separar información de color de luminancia y obtener resultados coherentes, ya que casi todos los métodos y modelos para segmentación en imágenes y video utilizan de manera implícita información de luminancia para lograr buenos resultados. Como ejemplos, KCNN aunque utilice el espacio HSV, este utiliza la información de valor para lograr una segmentación coherente, FO DPSO genera regiones con base a color, sin embargo, en los resultados también muestra que es muy sensible a cambios de luminancia.

Respecto al módulo V1 de PBSeg, RESOMV1 la capa de entrada realiza un modelado de fondo, pero bajo un enfoque autoroganizado, es decir, con base al estímulo de entrada se organizan los pesos. Esto es plausible con la teoría desarrollada por los modelos LISSOM. Para ser plausible con las teorías de organización cortical, se simularon las columnas verticales con filtros de orientación, los cuales separan la información de color con base a la distribución espacial, causando una segmentación suave de textura.

Una vez que PBSeg separa la información en los LGN, que RESOMV1 aprenda el fondo, y los filtros de orientación separa la información en textura, se obtienen las características del escenario separadas en color-textura y en luminancia. La información de luminancia se utiliza para realizar una segmentación de forma, mientras que la información de color-textura se utiliza para la segmentación de objetos estáticos. Ambos forma y color-textura son procedados mediante las redes pulsantes SCM y GSCM respectivamente. Por lo tanto, PBSeg se caracteriza no solo porque su esquema de funcionamiento es neuroinspirado, también porque los resultados obtenidos son plausibles con los fenómenos de percepción, ya que la detección de formas es plausible con procesos de agrupamiento perceptual tal como se menciona en modelos como PG-LISSOM y RF-PCNN. Además la detección de objetos estáticos se realiza generando regiones mediante la sincronización de pulsos y definiendo el tamaño del vecindario para el procesamiento de cada elemento, lo cual es el mecanismo por el cual modelos como LEGION justifican su plausibilidad con los principios de la *Gestalt*.

La retroalimentación en V4 entre formas y objetos estáticos es la forma de relacionar la información proveniente de luminancia y color, tal como se plantea que funciona V4 en [149]. Esta retroalimentación entre formas y objetos estáticos, además de mejorar el resultado de la segmentación estática, se utiliza para generar conexiones hacia abajo que relacionen información de contornos con superficie, tal como sucede en modelos como LAMINART. Otro aspecto interesante de PBSeg es que el resultado de la segmentación depende de información construida en capas corticales altas a partir de información pasada y actual, como el promedio de la información de los STM $V_4(x,y)^{z_c,t}$ y $\Gamma_q(x,y)^t$; es decir, PBSeg es un método que basa sus resultados en un esquema cognitivo ya que $V_4(x,y)^{z_c,t}$ y $\Gamma_q(x,y)^t$ son básicamente resultados de un conocimiento adquirido por PBSeg que se utilizan para obtener el resultado de segmentación estática.

En resumen, PBSeg es un método neurocognitivo para percepción de objetos que de acuerdo a los resultados de sección 5.3 fue aplicado exitosamente en segmentación de objetos estáticos considerando su coherencia en el tiempo y su capacidad para detectar objetos en escenarios obscuros, sombras. Adicionalmente, debido a que este método neuroinspirado genera información como luminancia-contornos y color-textura, puede no solo utilizarse para segmentar objetos estáticos en secuencias de video, también puede utilizarse en diferentes tareas relacionadas con análisis de video como:

- Detección de bordes en escenarios variantes en el tiempo.
- Simulación de respuesta cortical y procesos perceptuales en secuencias de video con escenarios reales. Este punto es una aportación interesante de PBSeg ya que la mayor parte de los modelos neurocomputacionales recurren a escenarios artificiales o ideales para comprobar sus teorías, mientras que PBSeg a pesar de ser un modelo inspirado de modelos neurocomputacionales, puede procesar escenarios reales.
- Se pueden cambiar algunos criterios de GSCM para generar un método de atención visual como el de [18].

Como conclusiones generales de los métodos de segmentación estática, la aportación de KCNN es que es un método exitoso para segmentación de objetos estáticos, mientras que la aportación de PBSeg es que es un método flexible para diferentes aplicaciones relacionadas al análisis neuroinspirado de objetos en secuencias de video.

Como trabajo futuro, se pretende realizar una versión de PBSeg que no tome como criterio que la información de formas está correctamente modelada, sino que a través de mediciones de coherencia entre formas y color se obtenga el mejor resultado posible en la segmentación de objetos estáticos. Esto se consideró desde un inicio, no obstante, estas mediciones de coherencia llevan una gran cantidad de cuadros y los videos disponibles no tienen suficiente información para realizar este proceso ya que al no existir un criterio de que la información está correctamente modelada y debido a los cambios presentes en un escenario, el proceso de medición de coherencia entre formas y color puede llevar miles de cuadros. En la práctica miles de cuadros equivale a unos cuantos minutos, por lo que este proceso puede ser útil en diferentes aplicaciones pero no se cuenta actualmente con una base de datos que se preste para los experimentos necesarios.

VI. DISCUSIÓN Y CONCLUSIÓN

Como se mencionó en el capítulo I, el objetivo de la tesis consiste en desarrollar un conjunto de métodos para segmentación de objetos con base a un esquema neuroinspirado. Una vez realizada la tesis, se realizaron diversas contribuciones relacionadas con el estado del arte de los modelos neurocomputacionales, desarrollo de nuevos modelos neuroinspirados y en segmentación en análisis de videos.

6.1. Estado del arte de modelos neurocomputacionales

Aunque en la literatura relacionada a las neurociencias existen muchos modelos que tratan de explicar el funcionamiento de la corteza visual y ciertos fenómenos perceptuales, no existe una clasificación de los modelos derivados del estudio de la corteza visual. Es decir, existen diversas publicaciones en artículos, libros o reportes donde se documenta el funcionamiento de la corteza visual de ciertos mamíferos como en **[**160**]**, del ser humano como en [142], modelos neurocomputacionales basados en la corteza visual como en [34], o documentación de RNA basadas en el comportamiento de ciertas neuronas de dicha corteza como la PCNN [66]. Sin embargo, no se reporta en la literatura un estudio de los modelos más relevantes inspirados en el funcionamiento de la corteza visual como el planteado en el capítulo II, donde se documentan desde modelos neurocomputacionales tan complejos como LAMINART hasta RNA como la PCNN. En este estudio los modelos se clasificaron en modelos basados en la Teoría de Resonancia Adaptiva, Modelos Auto-organizados, Redes Neuronales Pulsantes y Modelos Cognitivos.

Los modelos basados en la Teoría de Resonancia Adaptiva, son aquellos que describen el comportamiento de las capas de la corteza visual cuando estas efectúan tareas perceptuales como la estereopsis (definida en el capítulo II), percepción de movimiento, objetos, etc. Un aspecto interesante de estos modelos es que describen el comportamiento cortical para formar fenómenos perceptuales de varias capas de la corteza visual a un nivel de descripción de la dinámica celular. Los modelos Auto-organizados describen diversos mecanismos perceptuales bajo la hipótesis que las capas más bajas de la corteza visual se organizan con base a estímulos visuales. Estos modelos son muy útiles tanto para el desarrollo de modelos neurocomputacionales, como aplicaciones en procesamiento de información. Los modelos de redes pulsantes que están divididos en redes oscilatorias y

redes integra-dispara. Estos modelos se basan en que las neuronas generan como salida pulsos que se codifican de forma temporal y se utilizan para generar mecanismos de percepción similares a los estudiados en la teoría de la *Gestalt*. Los modelos cognitivos se enfocan a explicar los procesos que definen el pensamiento y la percepción a través del conocimiento adquirido. Un aspeco interesante del modelo de los Campos de modelado neuronal (NMF) vistos en el capítulo II es que tienen una capa de pesos que se va organizando en función al estímulo de entrada de manera similar a los modelos auto-organizados.

Mediante el análisis realizado, se pudo concluir que estos modelos aunque tienen diferencias significativas en su funcionamiento, los resultados pueden explicar fenómenos perceptuales similares. LAMINART y PG-LISSOM tienen en común que explican el agrupamiento perceptual como una integración de bordes en V1. Todas las redes pulsantes generan un proceso de sincronización neuronal que es consistente con ciertos principios de percepción de la *Gestalt*. PG-LISSOM es un modelo que mediante auto-organización de pesos realiza una codificación de pulsos que a su vez se utiliza para generar el fenómeno de agrupamiento perceptual.TLISSOM es el único modelo neuromputacional que se enfoca a la percepción de color, pero se puede equiparar a la relación bordes-superficies de LAMINART ya que este modelo define la percepción de color mediante la interacción de los mapas de OR y CR, donde OR se realiza a partir de la información de luminancia que a su vez se utiliza para bordes y CR se utiliza para el análisis de color en las superficies de los objetos. FORMOTION es el único modelo que se enfoca a explicar la percepción de movimiento, el cual es equiparable a diferentes teorías relacionadas a este fenómeno de percepción, ya que mediante información de cambios de luminancia y formas genera en la corteza medio-temporal superior (SMT) un modelo de atención o alerta que descarta aquellos patrones de movimiento que son irrelevantes y se enfoca en encontrar los objetos de movimiento.

En conclusión, los modelos inspirados en la corteza visual que son utilizados para procesar información como las RNA pulsantes o la SOM o que son utilizados para simular procesos perceptuales como LAMINART, FORMOTION, LISSOM estudian y fundamentan diferentes aspectos de los fenómenos de percepción desde diferentes teorías del funcionamiento neuronal, pero obtienen resultados que se relacionan entre si. Por lo

tanto, estos modelos pueden ser una base para el desarrollo de nuevos esquemas y nuevas propuestas para estudiar la aplicación de la percepción visual en áreas como visión por computadora o robotica.

Otro aspecto que se pudo notar con los modelos neurocomputacionales es que parten de las mismas teorías que las RNA utilizadas para procesar información. Modelos como LAMINART y FORMOTION parten de resonancia adaptiva, LISSOM parte de los modelos auto-organizados como la SOM, modelos como *Wilson-Cowan* o distintos modelos integra y dispara son pulsantes como PCNN.

En las diferentes implementaciones de FORMOTION, SOMRM, RF-PCNN, LISSOM y TLISSOM se pudo obervar que el tiempo de procesamiento por iteración de cada modelo no excede un segundo, y a partir de ellos se pudieron realizar redes como RESOM y GSCM y métodos como PBSeg. Por lo tanto, al menos para estos modelos se puede comprobar la primera hipótesis realizada donde se argumentaba que a través de modelos para simulación cortical y neuronal se pueden desarrollar nuevos esquemas para visión por computadora que aplicarán mecanismos de percepción.

6.2. Nuevos modelos neuroinspirados

A lo largo de este trabajo, se dearrollaron interesantes modelos neuroinspirados para análisis de video los cuales son: RESOM, NTCNN, T4CNN, GSCM y RF de color. Estos modelos fueron implementados con tiempos de procesamiento bastante satisfactorios para aplicaciones en tiempo real como se puede observar en las Tablas 6.1 y 6.2. A continuación se abordan las conslusiones generales de dichos modelos.

Tabla 6.1 Tiempos de procesamiento en una Workstation Dell Presicion con procesador Intel Xeon y GPU GEOFORCE GTX 550Ti con videos de 120x160.

	Velocidad de procesamiento en fps		
Método/Modelo	**MATLAB®**	**C++ con CPU**	**C++ con CPU/GPU**
RESOM	165	500	340
NTCNN	230	55	1000
RF-SCM	25	332	815
SOM-CNN	60	50	220

Tabla 6.2 Tiempos de procesamiento en una Workstation Dell Presicion con procesador Intel Xeon y GPU GEOFORCE GTX 550Ti con videos de 240x320.

	Velocidad de procesamiento en fps		
Método/Modelo	**MATLAB®**	**CPU con C++**	**C++ con CPU/GPU**
RESOM	30	95	97
NTCNN	85	13	335
RF-SCM	8	77	240
SOM-CNN	15	12	45

Red RESOM: Esta red que se presentó en el capítulo III es una de las mayores contribuciones de este trabajo ya que se pudo aplicar para diversos propósitos y su modelo parte de la SOMRM, una red documentada en el estado del arte de los modelos de la corteza visual del capítulo II. Esta red se puede utilizar con pocas neuronas para aprender escenarios reales en aplicaciones relacionadas a modelado de fondo como se vio en los capítulos III y V. En el capítulo V, esta red se pudo utilizar para representar los mecanismos de auto-organización de V1. La flexibilidad en el uso de esta red se basa en los parámetros de la regla Hebbiana que le permiten a la red adaptar su aprendizaje de acuerdo a las necesidades del método que utilice dicha red. Sin embargo, RESOM puede ser más útil, ya que es posible cambiar la cantidad de neuronas y se pueden agregar funciones de activación no lineales en la respuesta de $\eta(i,j)$ (ver subsección 3.2.2). Estas modificaciones permitirían a RESOM ser una red más flexible para solucionar diferentes problemas relacionados con análisis de video.

De acuerdo a la Tabla 6.1 y 6.2 el *frame rate* de RESOM se presta para utilizarla en diferentes aplicaciones en tiempo real. La rapidez en el procesamiento de esta red se debe a que tiene cálculos muy simples y después del primer cuadro solo realiza una iteración por cuadro, en la cual se calcula solo la competición y se actualiza la regla Hebbiana. Estas operaciones se reducen a operaciones aritméticas simples de multiplicaciones y sumas, las cuales pueden ser realizadas de forma muy rápida una computadora actual con tecnología de multiprocesamiento. La cantidad de cálculos que realiza RESOM puede parecer muy grande, sin embargo, muchos algoritmos recursivos, iterativos o que utilizan información local de vecindarios por cada pixel pueden utilizar una mayor cantidad de cálculos que RESOM. Otros algoritmos requieren cálculos estadísticos, exponenciales y logarítmicos por pixel que consumen altos tiempos de procesamiento. RESOM aumenta sus cálculos en función de la cantidad de pixeles y la cantidad de neuronas a un ritmo de $O(KM)$, mientras

que múltiples modelos tienen un crecimiento exponencial al aumentar la cantidad de datos. Por lo tanto, la contribución de RESOM radica no solo en su flexibilidad para el aprendizaje, también por sus tiempos de procesamiento.

Red NTCNN: Esta red que se presentó en el capítulo III resultó ser una estrategía muy útil para binarización, ya que en las primeras iteraciones elimina el ruido para realizar la binarización en las iteraciones finales. El criterio que determinó el éxito de esta red es colocar el umbral globa, z_o en uno para que los pixeles en $I_e(x,y)^t$ que son mayores a cero sean uno en $Is(x,y)^t$, pero eliminando los pixeles de ruido con a_1 y a_2, los cuales son determinados de acuerdo a las necesidades del método que esta utilizando la NTCNN. Dicha red se implementa con un algoritmo iterativo que realiza una convolucion con $I_e(x,y)^t$ y un kernel de 5x5 en cada iteración. Este modelo tiene tiempos que permiten aplicaciones en tiempo real, pero de acuerdo a la Tabla 6.2, la velocidad de implementación de NTCNN en GPU, mejora significativamente, de manera que una umbralización normal por GPU tiene una diferencia poco significativa en tiempos de procesamiento comparada con un método de binarización tradicional. La velocidad de procesamiento es directamente proporcional al tamaño del cuadro o imagen a procesar, con un comportamiento similar a $O(K)$. Por lo tanto, NTCNN es un modelo que basandose en el surgimiento de nuevas tecnologías puede ser implementado como una forma de binarización que elimina ruido en cualquier método que analice video en tiempo real.

GSCM: Esta red neuronal pulsante es otra de las aportaciones interesantes de este trabajo de tesis, ya que al igual que RESOM se puede utilizar para diferentes aplicaciones y parte del análisis de literatura del capítulo II. La inspiración de esta red es el uso de redes pulsantes como PG-LISSOM, LEGION, red de *Wilson-Cowan*, la red oscilatoria de conexiones difusivas y PCNN para explicar fenómenos perceptuales. Una ventaja de esta red es que tiene pocos parámetros en comparación con el resto de las redes pulsantes, los cuales son $\alpha_F z$, $\alpha_E z$ y σ_{zc}. Al igual que todas las redes, GSCM genera pulsos que se van agrupando para formar grupos de neuronas que representan los objetos en el escenario. Sin embargo, al igual que la red de *Wilson-Cowan* y la red oscilatoria de conexiones difusivas, la información del vecindario que tiene cada neurona tiene una influencia gaussiana y depende de las características del escenario. De acuerdo a los modelos de la red de conexiones difusivas y el modelo de *Wilson-Cowan*, si W_{zc} tiene un comportamiento

gaussiano que depende de las condiciones del escenario, se pueden generar mecanismos perceptuales a los de la teoría de la *Gestalt*. Por lo tanto, GSCM es una red neuronal pulsante plausible con mecanismos perceptuales que puede utilizarse para segmentar objetos en escenarios reales, tiene pocos parámetros y se puede utilizarse en aplicaciones de tiempo real.

En diferentes pruebas de desempeño en tiempos realizadas con dos GSCM a cinco iteraciones y con un vecindario de 3x3, esta red tiene tiempos de procesamiento bastante satisfactorios, ya que en MATLAB® tiene una velocidad de procesamiento promedio de 44 fps, en CPU con C++ de 500 fps y en CPU/GPU con C++ fue de 700 fps en videos de 120x160. Con videos de 240x320, los tiempos de procesamiento son 15fps para MATLAB®, 122fps para CPU con C++ y 254fps para CPU/GPU con C++. La flexibilidad de uso de GSCM con los tiempos de procesamiento reportados se debe principalmente a que esta red parte de un modelo de PCNN simplificada que solo ajusta el parámetro σ_{zc} el cual se puede adaptar según la información del escenario o la imagen de entrada.

En diversos experimentos se pudo comprobar que GSCM puede utilizar diversos espacios de color, y no solo recurrir a los campos receptivos de color de PBSeg. Por lo tanto, aunque en esta tesis GSCM se utilizó únicamente en PBSeg, también se puede desarrollar con este modelo multiples aplicaciones para segmentación en imágenes y video, tal como se muestra en la Figura 6.1, donde se aplicó una GSCM para cada canal de color, obteniéndose el resultado de segmentación que se muestra.

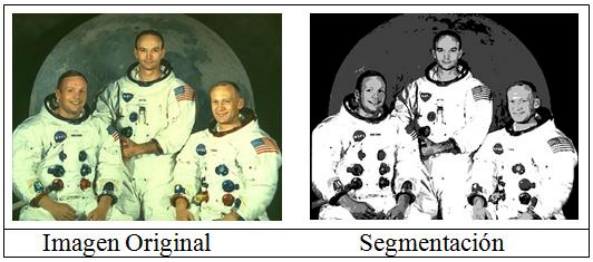


Figura 6.1. Ejemplo de segmentación utilizando únicamente GSCM con "*Training Image #323016*".

Campos Receptivos de Color: En el capítulo V se presentó un modelo que funciona como filtros selectivos de color a partir de las teorías y modelos de campos receptivos de color. Este modelo se basa en una DoG que en el caso de una imagen de un canal *I(x,y)* funciona adecuadamente. Con base a la teoría de funcionamiento de los campos receptivos de color,

se generó un modulo de LGN que funciona como un conjunto de filtros selectivos a diferentes colores. Esto es plausible con diversas teorías, ya que los LGN entre sus funciones descartan información.

Aunque estos filtros de color se utilizaron como el primer módulo en PBSeg, estos pueden utilizarse como modelos para diseñar métodos de segmentación para imágenes y video. Por ejemplo, en la Figura 6.2 se puede observar el resultado de un método de segmentación donde primero se obtiene la respuesta del modulo LGN, y con base a dicha respuesta, un conjunto de reglas realizan la segmentación. De la misma forma, se pueden diseñar diferentes métodos basandose en información de color y a la descomposición de color de los campos receptivos de color, donde un modulo similar a los LGN este acompañado de un conjunto de reglas de inferencia difusa, una red neuronal o cualquier clasificador analice la descomposición de color para realizar la segmentación.

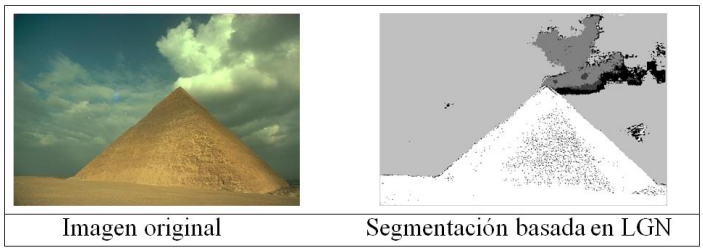


Figura 6.2. Resultados de segmentación de imagen utilizando como base los LGN.

6.3. Métodos de segmentación de objetos

La principal contribución de este trabajo fue el desarrollo de tres métodos para segmentar objetos estáticos y dinámicos, los cuales fueron: SOM-CNN, KCNN y PBSeg. Estos métodos contribuyeron a desarrollar nuevos esquemas en segmentación de objetos en secuencias de video, lo cual como se mencionó en el capítulo I, es el principal objetivo de la tesis. Dado que los capítulos III,IV y V se enfocan a describir dichos métodos, en esta sección se resumirán los aspectos más relevantes y concluyentes relacionados a SOM-CNN, KCNN y PBSeg.

SOM-CNN: Este método de segmentación de objetos dinámicos se enfoca a detectar objetos mediante modelado de fondo y un análisis estadístico de cambios de luminancia entre un cuadro y otro. Este funcionamiento parte de algunos modelos que describen la percepción de movimiento como un resultado de la retroalimentación de diferentes características como alerta visual y cambios de luminancia. En SOM-CNN se obtienen los

objetos que se encuentra en movimiento con el modelado de fondo basado en DR-SOM y la segmentación con NTCNN y el tercer módulo analiza estádisticamente los cambios de luminancia para ajustar a DR-SOM y la NTCNN.

Los resultados de SOM-CNN mostraron que el método logra detectar correctamente los objetos dinámicos en diferentes condiciones del escenario, pero tiene problemas de falsos negativos cuando existen fondos dinámicos, así como falsos positivos cuando un objeto de fondo se remueve. De acuerdo a las pruebas realizadas, SOM-CNN tiene muy buen desempeño respecto a los métodos de segmentación actuales en situaciones como fondos dinámicos, cámaras en movimiento, escenarios obscuros y de bajos contrastes. Sin embargo, en estas mismas pruebas se pudo observar que SOM-CNN tiene de los mejores desempeños en situaciones como objetos dinámicos que permanecen estacionarios y cambios de iluminación graduales y repentinos. El aspecto más relevante de SOM-CNN es el tiempo de procesamiento, ya que se logró obtener una velocidad en C++ de 250 fps en videos con tamaños de 120x160 y 45 fps en videos con tamaños de 240x320, mientras que el método más exitoso para detección de movimiento *Spectral-360* reporta 12 fps en videos con tamaño de 320x240.

KCNN: Este método fue iniciado en [**133**] y terminado en esta tesis; tuvo resultados muy exitosos en segmentación de objetos estáticos. De acuerdo a las pruebas realizadas con videos que tienen diferentes situaciones en el escenario, KCNN muestra resultados de segmentación que son coherentes a los objetos en escenarios que son variantes en el tiempo. Estos resultados revelan la contribución de KCNN, ya que la mayor parte de los métodos que segmentan imágenes pueden llegar a tener coherencia a ciertas características como color, textura, pero no necesariamente a los objetos presentes. Además, la mayor parte de los métodos de segmentación no consideran que un escenario pueda variar en el tiempo, de manera que al utilizarlos en una secuencia de video, no tienen coherencia siguiendo los cambios de los objetos, mientras que KCNN si la tiene. Dado que en la literatura no se reportan trabajos relacionados a segmentación de objetos estáticos en secuencias de video, entonces, de entre todos los métodos de segmentación de imágenes fijas, KCNN es el más exitoso en segmentar objetos en escenarios variantes en el tiempo.

El éxito de KCNN permite plantear un trabajo a futuro donde la segmentación de objetos dinámicos y estáticos se retroalimenten para mejorar el resultado de segmentación de objetos estáticos y dinámicos. En el capítulo I se planteó la hipótesis que la información de los objetos estáticos es tan útil como la de los dinámicos, y es a partir de KCNN que se puede iniciar trabajos basados en este planteamiento.

PBSeg: Este método que utiliza principios derivados de neurociencias y percepción visual para procesar la información de un escenario, mostró en diversas pruebas tener bastante flexibilidad para obtener una representación de formas de los objetos mediante contornos, así como una segmentación que es coherente a los objetos presentes en el escenario y sus cambios en el tiempo. Por lo tanto, también es un método exitoso para segmentar objetos en escenarios que son variantes en el tiempo, al igual que KCNN. Sin embargo, existen entre KCNN y PBSeg diferencias muy significativas en cuanto al funcionamiento. Una primera diferencia es que PBSeg brinda más información que KCNN, ya que se puede utilizar para diferentes propósitos en análisis del escenario. Una segunda diferencia es que este método tiene un mejor desempeño en escenarios obscuros y escenarios con sombras y reflejos que KCNN.

PBSeg plantea multiples trabajos futuros por su funcionalidad. Como se mencionó anteriormente, uno es mejorar la segmentación de objetos estáticos mediante la retroalimentación continúa de formas y color, en vez de asumir que la información de formas esta correctamente realizada. Otro es codificar mediante diferentes estrategías la información de los pulsos de GSCM para mejorar los resultados de segmentación estática. Otro aspecto importante de PBSeg, es que mediante los pulsos de la GSCM, los pulsos de SCM para detectar bordes, el agrupamiento perceptual y los resultados de los filtros de orientación, se puede obtener una gran cantidad de información del contenido del escenario que se puede utilizar no solo en segmentación de objetos estáticos, también se puede utilizar para diferentes aplicaciones en secuencias de video como atención visual, detectar alguna carcateristica de un objeto en específico o incluso en detección de movimiento. En conclusión, PBSeg es un método neuroinspirado que analiza un escenario variante en el tiempo obtenido en una secuencia de video que fue exitosamente aplicado en segmentación de objetos estáticos.

6.4. Conclusión general y trabajos futuros

Con base al estado del arte sobre modelos basados en la corteza visual y los resultados obtenidos por parte de los diversos modelos y métodos desarrollados en este trabajo de tesis, se puede concluir que es posible desarrollar trabajos relacionados al análisis de video a partir de esquemas perceptuales y modelos neuroinspirados con escenarios reales y en aplicaciones practicas que requieren velocidades de procesamiento en tiempo real. Sin embargo, para lograr resultados exitosos es necesario recurrir a otras metodologías que se derivan del mismo desarrollo del procesamiento de imágenes, tales como análisis estádisticos, conectividad entre pixeles vecinos, diseño de reglas de inferencia, separar el contenido de una imagen en clases, etc.

Recurrir a métodos neuroinspirados hibridos, es decir algorimtos basados en modelos neuroinspirados que utilizan elementos del procesamiento tradicional de imágenes, puede ser una forma muy novedosa y factible de diseñar estrategías para resolver problemas del área de visión por computadora ya que se puede generar soluciones similares a las habilidades de percepción. Por ejemplo, con base a los campos receptivos, PBSeg genera resultados invariantes a sombras y reflejos, debido a que logra aislar la información de color de luminancia sin tener pérdidas significativas de información como sucede con muchos espacios de color. Otro ejemplo de esta adaptabilidad perceptual es en el mismo PBSeg que mediante conexiones hacia abajo puede retroalimentarse por la gran cantidad de información del escenario que genera. Otro ejemplo es la manera en como SOM-CNN puede adaptarse a las condiciones del escenario, ya que aprende muy lentamente si un objeto dinámico permanece estacionario cientos de cuadros, pero puede aprender todo el escenario entre un cuadro y otro cuando una secuencia de video inicia o se genera un cambio de iluminación repentino. Estas formas de procesamiento de PBSeg y SOM-CNN se pueden obtener con otros métodos derivados del procesamiento clásico de imágenes, pero muchas veces con algoritmos complicados que son iterativos, recursivos o requieren una gran cantidad de cálculos por cada pixel presente en el cuadro, mientras que PBSeg y SOM-CNN recurren a modelos que se componen por una o dos ecuaciones simples que procesan la información del escenario según el estado de los parámetros. Estos modelos

acomodados de manera jerarquica, realizan un procesamiento mucho más rápido que otros algoritmos.

Un aspecto que contribuyó a implementar exitosamente los modelos y métodos en esta tesis es el manejo de tecnologías recientes, ya que aunque se trabajó con una *Workstation Dell Presision*, también se realizaron diferentes pruebas en una Laptop *Toshiba Satellite L735-SP3267CM* (modelo que salió al mercado en el año 2012), la cual tiene un procesador *Intel Core i5* y *Windows 7* como sistema operativo. En esta computadora se corrieron diferentes versiones de SOM-CNN, PBSeg, KCNN en MATLAB®, obteniéndose las mismas velocidades de procesamiento que en la *Workstation Dell Presision*. Las velocidades de procesamiento de estos métodos de segmentación son evidencia que SOM-CNN, PBSeg, KCNN se encuentran entre los métodos con mejor velocidad de procesamiento en la literatura. Además de esto, el soporte por GPU demostró que tecnologías que se utilizan en dispositivos embebidos también pueden soportar métodos neuroinspirados como los elaborados en esta tesis, ya que se logro con esta tecnología acelerar el procesamiento, obteniendo velocidades de procesamiento que superan incluso algunos de los mejores métodos de segmentación.

Como se mencionó previamente, trabajos futuros que se pueden realizar son muchos, ya que en la literatura analizada no se repotaron trabajos que analizan escenarios reales y variantes en el tiempo mediante modelos y métodos diseñados directamente de las teorías perceptuales y modelos neurocomputacionales. Sin embargo, los trabajos más próximos son los siguientes:

1. Mejorar los resultados de SOM-CNN con una versión distinta y más neuroinspirada de RESOM, una NTCNN con una plantilla de pesos *A* con ponderaciones asimétricas y que una RNA probailistica reeplace las inferencias del tercer modulo y funcione de manera similar a la capa cortical SMT.
2. Realizar un método que se retroalimente de los resultados de KCNN y SOM-CNN para mejorar el resultado de segmentación de objetos dinámicos.
3. Desarrollar un nuevo modelo de PBSeg que contenga más conexiones hacia abajo que retroalimenten a los resultados de luminancia y color. En este nuevo modelo, se

puede recurrir a la estructuración de conceptos de manera similar a los modelos neurocognitivos.

Finalizando, debido al éxito en velocidades de procesamiento y resultados de segmentación en los modelos RESOM, NTCNN, GSCM, filtros de color basado en RF así como los métodos SOM-CNN, KCNN y PBSeg, se puede concluir que estos modelos y métodos neuroinspirados son factibles para aplicaciones relacionadas a procesamiento y análisis de video como seguridad, sistemas de vigilancia, robotica, medicina, industria, etc. Adicionalmente, con base a los resultados obtenidos, esta tesis muestra evidencia de que los modelos y métodos que parten directamente de los equemas neuroinspirados, perceptuales y cognitivos pueden en algunos casos generar mejores resultados en aplicaciones de análisis de video que aquellos que parten de esquemas clásicos.

VII. REFERENCIAS

[1] Teddy Ko, "A survey on behavior analysis in video surveillance for homeland security applications," in *Workshop on Applied Imagery Pattern Recognition* , 2008, pp. 1-8.

[2] Nicola Cottini, Massimo Gottardi, and Nicola Massari, "A 33 mW 64x64 Pixel Vision Sensor Embedding Robust Dynamic Background Subtraction for Event Detection and Scene Interpretation," *IEEE Journal of Solid State Curcuits*, vol. 48, no. 3, pp. 850-863, Mar 2013.

[3] Gokhan Koray Gultekin and Afsar Saranli, "An FPGA based high performance optical flow hardware design for computer vision aplications," *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 270-286, May 2013.

[4] Sankalita Saha, Neal K. Bambha, and Shuvra S. Bhattacharyya, "Design and implementation of embedded computer vision systems based on particle filters," *Elsivier, Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1203-1214, Nov 2010.

[5] Janardan Misra and Indranil Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Elsevier Neurocomputing*, vol. 74, no. 1-3, pp. 239-255, Dic 2010.

[6] Shital Raut, M Raghuvanshi, R Dharaskar, and Adarsh Raut, "Image Segmentation – A State-Of-Art Survey for Prediction," in *Int. Conf. on Advanced Computer Control*, 2009, pp. 420-424.

[7] Dubravko Culibrk, Oge Marques, Daniel Socek, Hari Kalva, and Borko Furht, "Neural Network Approach to Background Modeling for Video Object Segmentation," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1614-1627, Nov 2007.

[8] Graciela Ramírez Alonso and Mario I Chacón Murguía, "New Trends on Dynamic Object Segmentation in Video Sequences: A Survey," *RIEE&C, Revista de ingeniería Eléctrica, Eléctronica y Computación*, vol. 11, no. 1, pp. 29-42, Dic 2013.

[9] Juan A. Ramírez-Quintana, Mario I. Chacon-Murguia, and Jose F. Chacon-Hinojos, "Artificial Neural Image Processing Applications: A Survey," *Engineering Letters*, vol. 20, no. 1, pp. 68-80, feb 2012.

[10] Liyuan Li, Weimin Huang, Irene Y H Gu, and Qi Tian, "Foreground Object Detection in Changing Background Based on Color Co-Occurrence Statistics," in *Proceedings on IEEE Sixth workshop on aplications of computer vision*, 2002, pp. 269-274.

[11] Di Fan, Maoyong Cao, and Changzhi Lv, "An Updating Method of Self-adaptive Background for Moving Objects Detection in Video," in *International Conference on Audio, Language and Image Processing*, 2008, pp. 1497-1501.

[12] Paul Withagen, Klamer Schutte, and Frans Groen, "Probabilistic Classification between Foreground Objects and Background," in *Proceedigs of the 17th International Conference on Pattern Recognition*, 2004, pp. 31-34.

[13] Lucia Maddalena and Alfredo Petrosino, "A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168-1177, Julio 2008.

[14] Mario I Chacon-Murguia, Sergio Gonzalez-Duarte, and Javier Vega P., "Simplified SOM-Neural Model for Video Segmentation of Moving Objects," in *Proceedings of International Joint Conference on Neural Networks*, Atlanta, Georgia, 2009, pp. 14-19.

[15] Mario I Chacon-Murguia and Sergio Gonzalez-Duarte, "An Adaptive Neural-Fuzzy Approach for Object Detection in Dynamic Backgrounds for Surveillance Systems," *IEEE Transactions on industrial electronics*, vol. 59, no. 8, pp. 3286-3298, Ago 2012.

[16] Alberto Faro, Daniela Giordano, and Concetto Spampinato, "Evaluation of the Traffic

Parameters in a Metropolitan Area by Fusing Visual Perceptions and CNN Processing of Webcam Images," *IEEE Transactions on neural networks*, vol. 19, no. 6, pp. 1108-1129, Jun 2008.

[17] Giovanni Costantini, Daniele Casali, Massimo Carota, and Renzo Perfetti, "A CNN-based Algorithm for Moving Object Detection in Stereovision Applications," in *European cConference on Circuit Theory and Design*, 2007, pp. 500-503.

[18] Dongyue Chen, Liming Zhang, and Juyang (John)Weng, "Spatio–Temporal Adaptation in the Unsupervised Development of Networked Visual Neurons," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 992-108, Jun 2009.

[19] Muhammad Ishtiaq, Arfan Jaffar, Ayaz Hussain, Abdul Basit, and Anwar M. Mirza, "Wavelet Based Video Segmentation using Self Organizing Map Neural Network," in *International Association of Computer Science and Information Technology - Spring Conference*, 2009, pp. 122-125.

[20] Esteban J. Palomo, Enrique Domínguez, Rafael M. Luque-Baena, and José Muñoz, "Image Compression and Video Segmentation Using Hierarchical Self-Organization," *Springer, Neural Processing Letters*, vol. 37, no. 1, pp. 69–87, Feb 2013.

[21] M. Egmont-Petersen and H. Handels D. Ridder, "Image processing with neural networks—a review," *Elsevier, Pattern Recognition*, vol. 35, no. 10, pp. 2279-2301, Oct 2002.

[22] Juan A Ramírez-Quintna and Mario I Chacón-Murguía, "Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década," *RIEE&C Revista de Ingeniería Eléctrica, Eléctronica y Computación*, vol. 9, no. 1, pp. 7-16, Jul 2011.

[23] Steven Yantis, *Visual Perception*, Psychology Press, Ed. USA: Philadelphia, 2001.

[24] Vicky Bruce, Patric Green, and Mark Georgeson, *Visual Perception Physiology Phsicology and Ecology*, 3rd ed. Sussex, Uk: Phsicology Press, 1996.

[25] Juan Jose Lopez-Ibor, Tomas Lopez-Alonso, and Maria Ines Lopez-Ibor, *lecciones de Psicología Medica*. Barcelona, 1999: Masson, 1999.

[26] Gilberto Gerardo Oviedo, "La definición del concepto de percepción en psicología con base a la teoría de la Gestalt," *Revista de estudios sociales*, vol. 18, no. 1, pp. 89-96, Aug 2004.

[27] Jorge Luis Martín, "Implicaciones epistemológicas de la noción de la forma en la psicología de la Gestalt," *Revista de historia de la psicología*, vol. 31, no. 4, pp. 37-50, Dic 2010.

[28] Patricia S Churchland, Christof Koch, and Terrence J Sejnowski, *What is computational neuroscience*. Cambridge, MA, USA: MIT Press, 1993.

[29] Mikhail I Rabinovich, Karl J Friston, and Pablo Varona, *Principles of Brain Dynamics*. Cambridge, MA, USA: MIT Press, 2012.

[30] Terrence J Sejnowsky, Koch Christof, and Patricia Churchland, "Computational Neuroscience," *Science*, vol. 24, 1988.

[31] Peter Dayan and Lawrence F Abbot, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Boston, MA, USA: MIT Press, 2001.

[32] Takeo Watanabe Yuko Yotsumoto, "Defining a Link between Perceptual Learning and Attention," *PLoS Biol*, vol. 6, no. 8, 2008.

[33] David H Hubel, *Ojos, Cerebro y visión*, Servicios Publicaciones, Ed. España: Universidad de Murcia, 2000.

[34] Risto Miikkulainen, James A. Bednar, Yoonsuck Choe, and Joseph Sirosh, *Computational Maps in the Visual Cortex*, Springer Sciencies Media Inc, Ed. New York USA, 2005.

[35] Sarah N Blythe and John H Krantz, "A Mathematical Model of Retinal Receptive Fields Capable of Form & Color Analysis," *Impulse: The Premier Journal for Undergraduate*

Publications in the Neurosciences., vol. 1, no. 1, June 2004.

[36] Hidehiko Komatsu, "Mechanisms of central color vision," *Current Opinion in Neurobiology*, vol. 8, no. 4, pp. 503-508, Aug 1998.

[37] Cesar uturbia Vicario, *Neurología de la visión*, 2nd ed. Barcelona, España: UPC, 1999.

[38] Gail carperter and Stephen Grossberg, "Adaptive Resonance Theory," Massachusetts Institute of Thecnology, Boston, Massachusetts, Thecnical Report 2002.

[39] Robert A Wilson and Frank C Keil, *The MIT encyclopedia of the cognitive sciences*, primera ed. Massachusetts, USA: MIT Press, 1999.

[40] Stephen Grossberg, "Cortical Dynamics of Form Perception," in *Proceedings of the International Joint Conference on Neural Networks*, 2003, pp. 8-10.

[41] Yongqiang Cao and Stephen Grossberg, "A Laminar Cortical Model of Stereopsis and 3D Surface Perception: Closure and da Vinci Stereopsis," Department of Cognitive and Neural Systems and Center for Adaptive Systems, Boston University, Boston, MA USA, Technical Report CAS/CNS-TR-2004-007, 2004.

[42] Stephen Grossberg, "Visual Motion Perception," Department of Cognitive and Neural Systems, Boston University, Boston, MA, USA, Thecnical Report.

[43] J. Berzhanskaya, S. Grossberg, and E. Mingolla, "Laminar cortical dynamics of visual form and motion interactions during coherent object motion perception," Department of Cognitive and Neural Systems, Boston University, Boston, MA, USA, Thecnical Report 2007.

[44] Stephen Grossberg, "View-Invariant Object Category Learning, Attention, Recognition, Search, and Scene Understanding ," in *Proceedings of International Joint Conference on Neural Networks*, Atlanta Georgia USA, June 14-19 2009, pp. 1002-1004.

[45] Stephen Grossberg and Tsung-Ren Huang, "ARTSCENE: A Neural System for Natural Scene Classification," CAS/CNS Technical Report 2007-017, 2007.

[46] DeLiang Wang and David Terman, "Locally Excitatory Globally Inhibitory Oscillator Networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 283-286, Jan 1995.

[47] Mauro Ursino, Elisa Magosso, and Cristiano Cuppini, "Recognition of Abstract ObjectsVia Neural Oscillators: Interaction Among Topological Organization, Associative Memory and Gamma Band Synchronization," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 316-335, Feb 2009.

[48] Naeem Shareef, DeLiang L. Wan, and Roni Yagel, "Segmentation of Medical Images Using LEGION," *IEEE Transactions on Medical Imaging*, vol. 18, no. 1, pp. 74-91, Jan 1999.

[49] P. Belardinelli, A. Mastacchi, V. Pizzella, and G.L. Romanii, "Applying a Visual Segmentation Algorithm to Brain Structure MR Images," in *IEEE Proceedings of the 1st International Conference on Neural Engeneering*, 2003, pp. 507-510.

[50] Jiangye Yuan, DeLiang Wang, Bo Wu, Lin Yan, and Rongxing Li, "Automatic Road Extraction from satellite imagery using LEGION Networks," in *Proceedings on International Conference on Neural Networks*, Atlanta Georgia, Jun 2009, pp. 14-19.

[51] Erdogan Çesmeli and DeLiang Wang, "Texture Segmentation Using Gaussian–Markov Random Fields and Neural Oscillator Networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 394-404, March 2001.

[52] Dênis Fernandes, Jeferson Polidoro Stedile, and Philippe Olivier Alexandre Navaux, "Architecture of Oscillatory Neural Network for Image Segmentation," in *Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing*, 2002, pp. 29-36.

[53] Denis Fernandes and Philippe Olivier Alexandre Navau, "A Low Complexity Digital

Oscillatory Neural Network for Image Segmentation," in *Proceedings IEEE International Symposium on Signal Processing and Information Technology*, 2004, pp. 365 – 368.

[54] Daniel Fernández et al., "Mismatch-tolerant CMOS oscillator and excitatory synapse for bioinspired image segmentation," in *IEEE International Symposium on Circuits and Systems*, 2005, pp. 4114-4117.

[55] Jordi Cosp and Jordi Madrenas, "Scene Segmentation Using Neuromorphic Oscillatory Networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1278-1296, Sep 2003.

[56] Ke Chen and DeLiang Wang, "Perceiving Geometric Patterns: From Spirals to Inside–Outside Relations," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1084-1102, Sep 2001.

[57] Guoshen Yu and Jean-Jacques Slotine, "Visual Grouping by Neural Oscillator Networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1871-1884, Dec 2009.

[58] Liang Zhao, Fabricio A. Breve, Marcos G. Quiles, and Roseli A. F. Romero, "Visual Selection and Shifting Mechanisms Based on a Network of Chaotic Wilson-Cowan Oscillators," in *Third International Conference on Natural Computation (ICNC 2007)*, 2007, pp. 754-762.

[59] Wolfwang Mass, "Networks of Spiking Neurons: The Third Generation of Neural Network Models," *Elsevier Sciences Neural networks*, vol. 10, no. 9, pp. 1659-1671, 1997.

[60] N.G. Pavlidis, D.K. Tasoulis, V.P. Plagianakos, and M.N. Vrahatis, "Spiking Neural Network Training Using Evolutionary Algorithms," , International Joint Conference on Neural Networks (IJCNN'05), 2005, pp. 2190 - 2194.

[61] Eugene M. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569-1572, Nov 2003.

[62] Timote Masquelier and Simon J. Thorpe, "Learning to recognize objects using waves of spikes and Spike Timing-Dependent Plasticity," in *International Joint Conference on neural Networks*, 2010, pp. 1-8.

[63] Helene Paugam-Moisy, "Spiking Neuron Networks: A Survey," Research Report 2006.

[64] José Antonio Pérez-Carrasco et al., "Fast vision through frameless Event-based sensing and convolutional processing application to texture recognition," *IEEE Transactions on Neural Network*, vol. 21, no. 4, pp. 609-620, April 2010.

[65] Christoph Rashe, "Neuromorphic Excitable Maps for Visual Processing," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 520-529, March 2007.

[66] Zhaobi Wang, Yide Ma, Feiyan Cheng, and Lizhen Yang, "A Review of pulse-coupled neural networks," *Elsevier image and vision computer*, vol. 28, no. 1, pp. 5-13, Ene 2010.

[67] Mario I Chacon and Jessica A. Mendoza, "A PCNN-FCM Time Series Classifier For Texture Segmentation," in *Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, El Paso USA, March 2011, pp. pp 1-6.

[68] H. Burton, "Visual Cortex Activity in Early and Late Blind People," *The Journal of Neuroscience*, vol. 23, no. 10, pp. 4005-4011, Mayo 2003.

[69] Kevin Brohan, Kevin Gurney, and Piotr Dudek, "Using reinforcement learning to guide the development of self-organised feature maps for visual orienting," in *Proceedings of the International conference on Artificial Neural Networks: Part II*, 2010, pp. 180-189.

[70] Minho Lee, Sang-Woo Ban, Jun-Ki Cho, Chang-Jin Seo, and Soon Ki Jung, "Modeling of Saccadic Movements Using Neural Networks," in *Proceedings on International Join Conference Neural Networks*, 1999, pp. 2386 – 2389.

[71] Alessio Plebe, "Learning Visual Invariance," in *Proceedings European symposium on*

artificial neural networks, 2006, pp. 73-76.

[72] Ravishankar Rao and Youping Xiao, "A computational model of early visual cortex using konio-cellular pathway projections," in *Proceedings on International join Conference on Neural Networks*, 2012, pp. 1-8.

[73] Yok-Yen Nguwi and Siu-Yeung Cho, "Two-tier Self-Organizing Visual Model for Road Sign Recognition," in *Proceedings on International Join Conference Neural Networks*, 2008, pp. 794-799.

[74] Gaëtan Martens, Chris Poppe, Peter Lambert, and Rik Van de Walle, "Unsupervised Texture Segmentation and Labeling Using Biologically Inspired Features," in *Proceedings IEEE Workshop on Multimedia Signal Processing*, 2008, pp. 159-164.

[75] Yuheng Wang and Roger S Gaborski, "Computational Modeling of Topographic Arrangements in Human Visual Cortex," in *World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2012, pp. http://elrond.informatik.tu-freiberg.de/papers/WorldComp2012/IPC2619.pdf.

[76] Ivonne Maricela Avila-Mora and Claudio Castellanos-Sanchez, "Bio-inspired clustering of moving objects," in *Proceedings on Global Congress on Intelligent Systems*, 2009, pp. 58-62.

[77] H. U. Bauer, M. Riesenhubert, D. Brockmann, and and T. Geisel, "Analysis of SOM-Based models for the development of visual maps," in *Procedings on WSOM*, 1997.

[78] D. Brockmann, H. -U. Bauer, M. Riesenhuber, and T. Geisel, "SOM-Model for the development of oriented receptive fields and orientation maps from non-oriented ON-center OFF-center inputs," in *Lecture Notes in Computer Sciences*, 1997, pp. 207-212.

[79] Choonseog Park, Yoon H. Bai, and Yoonsuck Choe, "Tactile or Visual?: Stimulus Characteristics Determine Receptive Field Type in a Self-organizing Map Model of Cortical Development," in *EEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing*, Nashville TN, 2009, pp. 6-13.

[80] Leonid I. Perlovsky, "“Vague-to-Crisp” Neural Mechanism of Perception," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1363-1367, Ago 2009.

[81] Leonid I. Perlovsky, "Neural Mechanisms of the Mind, Aristotle, Zadeh, and fMRI," *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 718-733, May 2010.

[82] Leonid I. Perlovsky, *Neural Networks and Intellect: Using Model-Based Concepts*, Primera ed. New York, USA: Oxford Univertity Press, 2001.

[83] James A. Bednar, "The Role of Internally Generated Neural Activity in Newborn and Infant Face Preferences," in *Face Perception in Infancy and Early Childhood*, O. Pascalis and A. Slater, Ed. New York, USA: Nova Science Publishers, 2002, pp. 133-142.

[84] Steven L Bressler, Craig G Richter, Yonghong Chen, and Mingzhou Ding, "Top-Down Cortical Influences in Visual Expectation," in *IEEE International Joint Conference on Neural Networks*, Vancouver, 2006, pp. 384-390.

[85] Matthew Luciu and Juyang (John) Weng, "Top–Down Connections in Self-Organizing Hebbian Networks: Topographic Class Grouping," *IEEE Transactions on autonomus mental development*, vol. 2, no. 3, pp. 248-261, Sep 2010.

[86] Andrew P Papliński and Lennart Gustafsson, "Feedback in Multimodal Self-organizing Networks Enhances Perception of Corrupted Stimuli," *Advances in Artificial Intelligence*, vol. 4304, pp. 19-28, 2006.

[87] Ming-Jung Seow, Ann Theja Alex, and Vijayan K. Asari, "Learning embedded lines of attraction by self organization for pose and expression invariant face recognition," *Optical Engineering*, vol. 51, no. 10, Oct 2012.

[88] Dempsey Chang, Keith V. Nesbitt, and Kevin Wilkins, "The Gestalt Principle of

Continuation Applies to both the Haptic and Visual Grouping of Elements," in *IEEE Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2007, pp. 15-20.

[89] Xurui Tan and Hao Tang, "Inquiry into the Shape Design of Display Space Based on Visual Perception," in *International Conference on Computer-Aided Industrial Design & Conceptual Design*, 2010, pp. 737 - 741.

[90] Yongzhen Huang, Kaiqi Huang, Dacheng Tao, Tieniu Tan, and Xuelong Li, "Enhanced Biologically Inspired Model for Object Recognition," *IEEE Transactions on systems, man and cybernetics: Part B-Cybernetics*, vol. 41, no. 6, pp. 1668-1680, Dec 2011.

[91] Hou Minfeng and Fu Liming, "On Auto Bionic Modeling Design Study Based on Cognitive Psychology Theories," in *International Conference on Mechatronic Science, Electric Engineering and Computer*, 2011, pp. 1641-1644.

[92] Yansheng Ming, Hongdong Li, and Xuming He, "Connected Contours: a New Contour Completion Model that Respects the Closure Effect," in *IEEE Conference on Computer Vision and Pattern Recognition* , 2012, pp. 829-836.

[93] Bo Yu and Liming Zhang, "Pulse-coupled neural networks for countour and motion matching," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, Sep 2004.

[94] Lin Wei-Song, Liu An-Te, and Fang Chun-Hsiung, "Computational Autonomus visual perception using cellular neural networks," in *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Giardini Naxos, Italia, 2005, pp. 198-202.

[95] Culibrk Dubravko, Oge Marques, and Daniel Socek, "Neural Network Approach to Background Modeling for Video Object Segmentation," *IEEE Transactions On Neural Networks*, vol. 18, no. 6, pp. 1614-1627, Nov 2007.

[96] Afsane Ghasemi and Reza Safabakhsh, "Unsupervised Foreground-Background Segmentation Using Growing Self Organizing Map in Noisy Backgrounds," in *3rd International conference on Computer Research and Development*, 2011, pp. 334-338.

[97] Volker Baier, "Motion Perception with Recurrent Self-Organizing Maps Based Models," in *Proceedings of International Joint Conference on Neural Networks*, Montreal Canada, 2005, pp. 1182-1186.

[98] Mario I. Chacon-Murguia and Jesus David Urias-Zavala, "A Comparison Between a DTCNN and SOM like Approach for Dynamic Object Detection in Videos," in *North American Fuzzy Information Processing Society*, 2012, pp. 1-6.

[99] Zhong-Lin Lu and George Sperling, "Three-systems theory of human visual motion perception: review and update," *Journal of the Optical Society of America A Optical, Image Science, and Vision*, vol. 19, no. 10, pp. 2331-2370, 2002.

[100] Xiaoya Yu, Jinglong Wu, and Shengfu Lu, "A Survey of Brain Mechanisms of Visual Motion Perception," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, 2013, pp. 1789-1792.

[101] Dominique Ravier, "A Service Oriented Framework Architecture for Intelligent Video Surveillance Systems," in *Fifth International Conference on Digital Telecommunications*, 2010, pp. 123-127.

[102] Teddy Co, "A Survey on Behavior Analysis in Video Surveillance for Homeland Security Applications," in *Workshop Applied Imaginery Pattern Recognition*, 2008, pp. 1-8.

[103] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers, "Wallflowers: Principles and Practice of Background Techniques for Video surveillance," in *International Conference on Computer Vision*, 1999, pp. 255-261.

[104] Sebastian Brutzer, Benjamin Höferlin, and Gunther Heidemann, "Evaluation of Background Subtraction Techniques for Video Surveillance," in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1937-1944.

[105] Byun Park, "A unified approach to background adaptation and initialization in public scenes," *Elsevier Pattern Recognition*, vol. 46, no. 7, pp. 1985-1997, Julio 2013.

[106] Wang Zhiming and Bao Hong, "Cooperative Neural Network Background Model for Multi-Modal Video Surveillance," in *Seventh International Conference on Computational Intelligence and Security*, 2011, pp. 249-254.

[107] Paul Withagen, Klamer Schutte, and Frans Groen, "Probabilistic Classification between Foreground Objects and Background," in *Proceedigs of the 17th International Conference on Pattern Recognition*, 2004, pp. 31-34.

[108] Xiaoodong Cai, Falah Ali, and E Stpidis, "Backgrund modeling for detecting move-then-stop arbitrary-long time video objects," in *Workshop on Image analysis for multimedia interactive services*, 2009, pp. 197-200.

[109] Fan-Chieh Cheng, Shih-Chia Huang, and Shanq-Jang Ruan, "Illumination-Sensitive Background Modeling Approach for Accurate Moving Object Detection," *IEEE Transactons on broadcasting*, vol. 57, no. 4, pp. 794-801, Dec 2011.

[110] Fan-Chieh Cheng, Shih-Chia Huang, and Shanq-Jang Ruan, "Scene Analysis for Object Detection in Advanced Surveillance Systems Using Laplacian Distribution Model," *IEEE Transactions on systems, man and cybernetics: Part C: Applications and review*, vol. 41, no. 5, pp. 589-598 , Sep 2011.

[111] Jithendra K Paruchuri, Edwin P Sathiyamoorthy, and Chung-Hao Chen Sen-ching S Cheung, "Spatially Adaptive Illumination Modeling for Background Subtraction," in *IEEE Int Conference on Computer Vision*, 2011, pp. 1745-1752.

[112] Traiwit Intachak and Watcharin Kaewapichai, "Real-Time Illumination Feedback System for Adaptive Background Subtraction Working in Traffic Video Monitoring," in *Int symposium on Intelligent signal processing and communications*, 2011, pp. 1-5.

[113] Mohamed Salah Salhi, Najet Arous, and Noureddine Ellouze, "Principal temporal extensions of SOM: Overview," *Int Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 2, no. 4, pp. 61-84, Dic 2009.

[114] Koller Daphne and Nir Friedman, "Background material," in *Probabilistic Graphical Models: principles and techniques*. Cambrige, Massachusetts, London, England: MIT Press, 2009, ch. A, p. 1144.

[115] Juan A Ramirez-Quintana and Mario I Chacon-Murguia, "Self-Organizing Retinotopic Maps Applied to Background Modeling for Dynamic Object Segmentation in Video Sequences," in *Aceptado para International Joint Conference on NeuralNetworks*, 2013.

[116] Leon Chua and Tamas Rosca, *Cellular neural networks and visual computing Foundation and applications*, Primera ed. New York, USA: Cambridge University Press, 2004.

[117] Leon Chua and Lin Yang, "Cellular Neural Networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257-1272, Oct 1988.

[118] C. Haiyang and M. Napolitano G.Yu, "A survey of optical flow techniques for UAV navigation applications," in *International Conference on Unmanned Aircraft Systems*, 2013, pp. 710-716.

[119] Mario I. Chacon-Murguía, Graciela Ramirez-Alonso, and Sergio Gonzalez-Duarte, "Improvement of a Neural–Fuzzy Motion Detection Vision Model for Complex Scenario Conditions," in *Int Conf on Neural Networks*, Dallas, 2013, pp. 1-8.

[120] Li Shuhua and Guo Gaizhi, "The application of improved HSV color space model in image

processing," in *2nd International Conference on Future Computer and Communication*, 2010, pp. 10-13.

[121] Youngeun An, Muhammad Riaz, and Jongan Park, "CBIR based on adaptive segmentation of HSV color space," in *12th International Conference on Computer Modelling and Simulation*, 2010, pp. 248-251.

[122] Su Ching-Hung, Chiu Huang-Sen, and Hsieh Tsai-Ming, "An efficient image retrieval based on HSV color space," in *International Conference on Electrical and Control Engineering*, 2011, pp. 5746-5749.

[123] Shamik Sural, Gang Qian, and Sakti Pramanik, "Segmentation and histogram generation using the HSV color space for image retrieval," in *Int Conf on Image Processing*, 2002, pp. 589-592.

[124] Rafael Gonzalez and Richard E Woods, *Digital Image Processing*, Segunda ed. New Jersey, USA: Pretince Hall, 2002.

[125] L Li W Huang, I Y H Gu, and Q Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Transactions on Image processing*, vol. 13, no. 11, pp. 1459–1472, Nov 2004.

[126] Juan A Ramirez-Quintana and Mario I Chacon-Murguia, "An Adaptive Unsupervised Neural Network based on Perceptual Mechanism for Dynamic Object Detection in Videos with Real Scenarios," *Springer Neural Processing Letters*, vol. Publisehd Online, pp. 1-25, Ago 2014.

[127] Ronald E Walpole, Raymond H Myers, and Sharon L Myers, *Probabilidad y estadística para ingenieros*, 6th ed. Mexio: Pretince Hall, 1999.

[128] Lucia Maddalena and Alfredo Petrosino, "The SOBS algorithm: what are the limits?," in *IEEE Computer Vision and Pattern Recognition Workshops*, Providence R.I., 2012, pp. 21-26.

[129] Ruben Heras Evangelio, "Complementary Background Models for the Detection of Static and Moving Objects in Crowded Environments," in *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, Klagenfurt, 2011, pp. 71-76.

[130] Shengping Zhang and Hongxun Yao, "Dynamic background modeling and subtraction using spatio-temporal local binary patterns," in *IEEE International Conference on Image Processing*, San Diego, 2008, pp. 1556 - 1559.

[131] Sedky Mohamed Hamed Ismail, Moniri Mansour, and Chibelushi Claude Chilufya, "Object Segmentation Using Full-Spectrum Matching of Albedo Derived from Colour Images," Staffordshire University, Patente No no. 0822953.6 16.12.2008 GB, Aplicación internacional PCT/GB2009/002829, 2008/2009.

[132] Yuanlong Yu, George K. I. Mann, and Raymond G. Gosine, "A Goal-Directed Visual Perception System Using Object-Based Top–Down Attention," *IEEE Transactions on autonomus mental development*, vol. 4, no. 1, pp. 87-103, March 2012.

[133] Jesus D Urias, "Estudio del modelos CNN en la segmentación de objetos estáticos y dinámicos en secuencias de video," Instituto Tecnologico de Chihuahua, Chihuahua, Tesis de maestría 2011.

[134] S Saraswathi and AAllirani, "Survey on Image Segmentation via Clustering," in *Int Conf on Information Communication and Embedded Systems*, 2013, pp. 331-335.

[135] S Sridevi and M. Sundaresan, "Survey of image segmentation algorithms on ultrasound medical images," in *Int Conf on Pattern Recognition, Informatics and Mobile Engineering*, 2013, pp. 215-220.

[136] DeLiang Wang, "A comparison of CNN and LEGION networks," in *International Join Conference on Neural Networks*, 2004, pp. 1735-1740.

[137] Takashi Inoue and Yoshifumi Nishio, "Applications of Color Image Processing Using Three-Layer Cellular Neural Network Considering HSB Model," in *Proceedings of International Joint Conference on Neural Networks*, Atlanta, Georgia, 2009, pp. 1135-1342.

[138] Anna Wilbik, "Cellular Neural Networks for Color Image Segmentation," in *Lecture Notes in Computer Science*, 2005, pp. 525-530.

[139] Inhye Yoon, Seonyung Kim, Donggyun Kim, Monson H. Hayes, and Joonki Paik, "Adaptive Defogging with Color Correction in the HSV Color Space for Consumer Surveillance System," *IEEE Transactions on Customer Electronics*, vol. 58, no. 1, pp. 111-116, Feb 2012.

[140] Choe Yoonsuck, "Perceptual Grouping in a Self-Organizing Map of Spiking Neurons," The University of Texas at Austin, Austin, TX, Disertación doctoral 2001.

[141] Frisby John and Stone James, *Seeing, The computational approach to biological vision*, Segunda ed. USA, Massachusetts: MIT Press, 2010.

[142] John P Frisby and James V Stone, *Seeing: The Computational Approach to Biological Vision*. Boston, MA, USA: MIT Press, 2010.

[143] Judah Ben De Paula, "Modeling the self-organization of color-selective neurons in the visual cortex," The University of Texas at Austin, Austin, TX, Tesis de doctorado 2007.

[144] James A Bednar, Judah B De Paula, and Risto Miikkulainen, "Self-organization of color opponent receptive fields and laterally connected orientation maps," in *Neurocomputoing*, 2005, pp. 1-6.

[145] James Bernard, "Building a mechanistic model of the development and function of the primary visual cortex," *Journal of Physiology*, vol. 106, no. 5-6, pp. 194–211, Sep 2012.

[146] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, "Color Transfer between Images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34-41, Sep/Oct 2001.

[147] Jonathan C Horton and Lawrence C Sincich, "A New Foundation for the Visual Cortical Hierarchy," in *The cognitive Neurosciences III*, Massachuetts Institute of Technology, Ed. Hannover, USA: MIT Press, 2004, ch. III, pp. 233-243.

[148] Margaret Livingstone and David Hubel, "Segregation of Form, Color, Movement, and Depth: Anatomy, Physiology, and Perception," *Science*, vol. 240, no. 4853, pp. 740-749, May 1988.

[149] Norbert Kruger et al., "Deep Hierarchies in the Primate Visual Cortex: What Can We Learn For Computer Vision?," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1-24, 2012.

[150] Yongqiang Cao and Stephen Grossberg, "Stereopsis and 3D Surface Perception by Spiking Neurons in Laminar Cortical Circuits: A Method for Converting Neural Rate Models into Spiking Models," Department of Cognitive and Neural Systems, Boston University, Boston, MA, USA, Thecnical Report SyNAPSE program, 2011.

[151] Kun Zhan, Hongjuan Zhang, and Yide Ma, "New Spiking Cortical Model for Invariant Texture Retrieval and Image Processing," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1980-1986, Dic 2009.

[152] Yuli Chen, Sung-Kee Park, Yide Ma, and Rajeshkanna Ala, "A New Automatic Parameter Setting Method of a Simplified PCNN for Image Segmentation," *IEEE Transactions on neural networks*, vol. 22, no. 6, pp. 880-892, Jun 2011.

[153] Hui Wei and Heng Wu, "A Neurocomputing Model for Ganglion Cell's Color Opponency Mechanism and Its Application in Image Analysis," in *International Joint Conference on Neural Networks*, 2013, pp. 574-581.

[154] Luis M Martinez and Jose-Manuel Alonso, "Complex receptive fields in primary visual cortex," *Neuroscientist*, vol. 9, no. 5, pp. 317-331, Oct 2003.

[155] Stephen L Chiu, "Fuzzy Model Identification Based on Cluster Estimation," *Journal of Intelligent and Fuzzy Systems*, vol. 2, pp. 267-278, 1994.

[156] Felzenszwalb P F and Huttenlocher D P, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167-181, Sep 2004.

[157] Couceiro Micael S, Rocha Rui P, N M Fonseca Ferreira, and J A Tenreiro Machado, "Introducing the fractional-order darwinian PSO," *Signal, Image and Video Processing*, vol. 6, no. 3, pp. 343-350, Sep 2012.

[158] Cao bin, Shen Xuanjing, and Qian Qingji, "Application of two-order particle swarm optimization algorithm in image segmentation," in *Int Conf on Computer-Aided Industrial Design*, 2010, pp. 749-752.

[159] N Gopi Raju and P A Nageswara Rao, "Particle Swarm Optimization Methods for Image Segmentation Applied In Mammography," *Int. Journal of Engineering Research and Applications*, vol. 3, no. 2, pp. 1572-1579, Nov-Dic 2013.

[160] Bertram Payne and Alan Peters, *The cat primary visual cortex*, Academic Press, Ed. San Diego,CA, USA: Boston University School of Medicine, 2002.

XIII. ÍNDICE DE TÉRMINOS Y NOTACIÓN

En esta parte se resumen todos los términos utilizados en la tesis. Aparecen todos los términos de casi todos los modelos y métodos de la tesis, excepto los modelos LAMINART, NMF y red de *Wilson Cowan*, los cuales se definieron en la tesis con base a sus respectivas referencias.

***k*:** Pixel.

***K*:** Número total de pixeles de una imagen o un cuadro.

x, y: Índices de posición espacial del pixel, mientras que *z* se refiere a un canal especifico.

***(x_1,y_1):** Ejemplo de un pixel cualquiera *(x,y)*.

***X, Y*:** Máximo número en *x,y*.

R, G, B, H, H_2, H_3, S, V, X_{hsv}, Y_{hsv}, Z_{hsv}: Canales de color rojo (R), verde (G), azul (B), matiz (H), matiz rotado (H_2 y H_3), saturación (S) y valor (V), X_{hsv}, Y_{hsv} y Z_{hsv} son coordenadas cartesianas del espacio HSV.

***R_g*:** Región.

***t*:** Índice de cuadros en secuencias de video, tiempo.

***$I(x,y,z)^t$*:** Cuadro de video original.

***$I_b(x,y,z)^t$*:** Cuadro de modelado de fondo.

***$I_e(x,y)^t$*:** Distancia euclidiana entre $I(x,y,z)^t$ e $I_b(x,y,z)^t$.

***$I_{eb}(x,y)^t$*:** Umbralización de $I_e(x,y)^t$.

***$I_{d8}(x,y)^t$*:** Regiones con conectividad 8 de $Is(x,y)^t$.

***$I_n(x,y)^2$*:** Imagen para medir condición de *bootstrapping*.

***$Is(x,y)^t$*:** Segmentación de objetos dinámicos con NTCNN.

***$IV(x,y,z,m_k)$*:** Diferencias absolutas de imágenes.

***$I_s(x,y)^t$*:** Segmentación auxiliar en cambios de iluminación con NTCNN.

***$Iv(x,y)^t$*:** Diferencia entre $I(x,y,z)^t$ y $I(x,y,z)^{t-1}$.

***σ*:** Desviación estándar del cuadro $I(x,y,z)^t$.

σ_{ch}: Desviación estándar de métricas de *changedetection*.

***μ*:** Media del cuadro $I(x,y,z)^t$.

μ_{ch}: Media de métricas de *changedetection*.

***μ_d*:** Promedio de μ_v en una ventana de tiempo.

$\boldsymbol{\mu_n}$**:** Media de $I_n(x,y)^t$.

$\boldsymbol{\mu_v}$**:** Media de $Iv(x,y)^t$.

$\boldsymbol{\mu_z}$: Media poblacional de métricas de *changedetection*.

E: Entropía del cuadro $I(x,y,z)^t$.

$\boldsymbol{L_v}$**:** Media de cantidad de regiones en $Iv(x,y)^t$.

$\boldsymbol{L_d}$**:** Promedio de L_v en una ventana de tiempo.

$\boldsymbol{E_v}$**:** Entropía de cantidad de regiones en $Iv(x,y)^t$.

$\boldsymbol{E_d}$**:** Promedio de E_v en una ventana de tiempo.

$\boldsymbol{E_p}$**:** Cambio de entropía.

$\boldsymbol{\alpha_o}$**,** $\boldsymbol{\sigma_o}$**,** $\boldsymbol{a_{1o}}$**,** $\boldsymbol{a_{2o}}$ **y** $\boldsymbol{s_2}$**:** Parámetros de SOM-CNN.

$\boldsymbol{\mu_{\mu d}}$ **y** $\boldsymbol{\mu L_d}$**:** Promedios de μ_d y L_d respectivamente.

$\boldsymbol{Th_1}$**,** $\boldsymbol{Th_2}$**:** Umbrales para distintos métodos.

$\boldsymbol{s_d(s_1,s_2,\mu_d)}$**:** Función de ajuste sigmoidal.

***DE*:** Nivel de entropía para cambiar iluminación.

$\boldsymbol{\Delta E}$**:** Diferencia de entropía.

***FlagLabel(id)*:** Vector de banderas.

id: Índice de bandera.

$\boldsymbol{z_{\alpha/2}}$**:** Nivel de confianza.

Índices de redes Neuronales y modelos neurocomputacionales

***i, j*:** Índices de neuronas artificiales.

***I, J*:** Máximo número en *i, j*.

***m*:** Neurona de RESOM.

***M*:** Máximo número de neuronas.

***a, b*:** Índice de campos receptivos de RGC a LGN.

***u, v*:** Índice de campo receptivo.

1. Redes auto-organizadas y DR-SOM.

$\boldsymbol{\eta(i,j)}$**,** $\boldsymbol{\eta(m)^z}$**:** Respuesta de neurona ganadora.

$\boldsymbol{W_{in}}$**:** Neurona ganadora SOM *Kohonen*.

***p,q*:** Índice de vecinos en neuronas de V1**.**

***(i_w,j_w)*:** Índice de posición de neurona ganadora.

***$\omega(k,m)^{z,n}$*:** Pesos de neuronas de la RESOM.

***$W(m)^{z,n}$*:** Neuronas de la RESOM.

***$V_w(x,y,z)^{m,t}$*:** Arreglo de imágenes RGB definido desde un conjunto de neuronas $W(m)^{z,n}$.

***$IV(x,y,z,m_k)$*:** Diferencia absoluta entre todas las imágenes del arreglo $V_w(x,y,z)^{m,t}$.

***A_v*:** Desfasamiento dado por la ecuación 3.23.

***L(u,v,a,b)*:** Campo receptivo.

***$\varsigma(a,b)$*:** Respuesta de LGN.

***$\psi(.)$*:** Función saturación.

***s(i,j)*:** Respuesta de V1 sin conexiones laterales.

***N_f*:** Iteraciones de la RESOM por cuadro.

***T_f*:** Cantidad de iteraciones para convergencia de la RESOM.

***n_L*:** Iteraciones de LISSOM.

***n_s*:** Iteraciones de SOMRM.

***$I_d(x,y)^t$*:** Cuadro de diferencias de las neuronas de la RESOM.

***A_L, E_L, I_L*:** Pesos en LISSOM aferentes, laterales estimuladores y laterales inhibidores.

***α*:** Taza de aprendizaje de la RESOM y SOMRM.

***α_s*:** Segunda taza de aprendizaje para evitar que α caiga a cero.

***$\beta(m)^z$*:** Función vecinal.

***$\beta(i,j)$*:** Función vecinal.

***σ_u*:** Amplitud de gaussiana de entrada.

***σ_β*:** Amplitud del radio de vecindario en RESOM.

***σ_c y σ_s*:** Amplitud de gaussianas para RF ON/OFF de RGC a LGN.

***σ_a y σ_b*:** Amplitud de gaussianas para filtro *Gabor*.

t_a, t_b: Índices auxiliares de cuadro.

2. Redes Neuronales Celulares (CNN).

***(i_N,j_N)*:** Índices de neuronas vecinales.

***N_r*:** Vecindario.

***n_c*:** Iteraciones de la CNN.

N_v**:** Vecindario.

***NC*:** Cantidad de iteraciones de la CNN por cuadro.

Th(i,j): Umbral definido por la NTCNN.

$\chi(i,j)^{n_c}$**:** Estado de la CNN.

$y_c(i,j)^{n_c}$**:** Salida de la CNN.

$Z_c(i,j)$**:** Desfasamiento de cada elemento.

***ac*:** Elementos de A.

***Nac*:** Producto punto de los vecinos de la entrada por los *ac*.

$A(i_N,j_N)$**:** Plantilla de pesos relacionada con la salida.

$B(i_N,j_N)$**:** Plantilla de pesos relacionada con la entrada.

$I_{bc}(x,y,z)^t$**:** Salida de 4TCNN.

$hbc(i_L,z)$**:** Matriz con los histogramas de los canales z de $I_{bc}(x,y,z)^t$.

a_1,a_2, z_o**:** Parámetros NTCNN.

$N_{r4}, N_8, N_{(25-8)}$**:** Vecindarios de la NTCNN.

4. Red Neuronal Pulso-Acoplada (PCNN).

n_p**:** Iteraciones de la PCNN.

$Fp(i,j)^{n_p}$**:** Alimentador (*Feeding*).

Vp_F**:** Constante de alimentador.

αp_F**:** Constante de retardo del alimentador.

$Lp(i,j)^{n_p}$**:** Encadenador (*Linking*).

Vp_L**:** Constante de encadenamiento.

αp_L**:** Constante de retardo del encadenador.

$Up(i,j)^{n_p}$**:** Agregación de las sinapsis.

$\Theta p(i,j)^{n_p}$**:** Función de umbral.

$Yp(i,j)^{n_p}$**:** Salida de la PCNN.

***Wp*:** Pesos de entrada para el alimentador.

***Mp*:** Pesos de entrada para el encadenador.

***SYp(i,j)*:** Suma de pixeles de $Yp(i,j)^{n_p}$.

***Wp*:** Pesos de la red neuronal SCM.

5. LEGION

I_L**:** Estímulo de entrada.

x_{Li}**:** Respuesta del oscilador y estimulador local.

y_{Li}**:** Respuesta del inhibidor global.

ξ**:** Constantes que caracteriza la caída de la tangente hiperbólica.

e, γ**:** Constantes de y_{Li}.

ρ_g**:** Amplitud del ruido gaussiano.

SS_i**:** Respuesta de los vecinos.

$H(.)$**:** Función de heaviside.

6. BPSeg.

Vor(i,j): Campos receptivos de orientación.

Ga_r**:** Gaussian de campo receptivo, donde r={c,s}.

RF(a,b): campo receptivo.

RG(a,b): Campo receptivo rojo/verde.

GR(a,b): Campo receptivo verde/rojo.

BY(a,b): Campo receptivo azul/amarillo.

sRF(a,b): Campo receptivo con convoluciones normalizadas.

SRF(a,b): Campo receptivo con ajuste en cero.

RFs(a,b): Campo receptivo normalizado.

RGON: Campo receptivo rojo-verde ON.

RGOF: Campo receptivo rojo-verde OFF.

GRON: Campo receptivo verde-rojo ON.

GROF: Campo receptivo verde-rojo OFF.

BYON: Campo receptivo azul-amarillo ON.

BYOF: Campo receptivo azul-amarillo OFF.

$RFO(x,y,\theta)$**:** Campo receptivo de orientación.

c***RFO*****:** Parámetro de ajuste de campo receptivo de orientación.

$\varsigma_R, \varsigma_Y, \varsigma_G, \varsigma_B$**:** Canales de color aferentes.

Lum(a,b)**:** Campos receptivos de luminancia.

$V_{mn}(x,y)^{z_o t}$**:** Capa de campos receptivos de orientación V1.

$V_m(x,y)^{z_c}$**:** Primer capa cortical en V1.

$V_n(x,y)^{z_n}$**:** Segunda capa cortical en V1.

$V_g(x,y)^{z_n}$**:** Tercer capa cortical de color en V1.

$V_t(x,y)^{z_n}$**:** Primera capa cortical de color en V2.

$V_{tb}(x,y)^{z_n}$**:** Segunda capa cortical de color en V2.

z_c**:** Canales corticales, donde $z_c = \{ \varsigma_R, \varsigma_Y, \varsigma_G, \varsigma_B, ON, OFF \}$.

$\Gamma_c(x,y)^t$**:** Tercer capa cortical de luminancia que contiene contornos.

$\Gamma_q(x,y)^t$**:** Agrupamiento perceptual.

$I_{rb}(x,t)^t$**:** Información de color en los campos receptivos.

Θ**:** Filtro de los LGN.

Θ_{z_c}**:** Ajustes para filtros de los LGN.

hCr: Histograma de mapa de color.

hI_{rb}**:** Histograma de campos receptivos de color.

hI_r**:** Histograma de matiz H_3 dividido en cuatro secciones.

hI_h**:** Histograma de matiz H dividido en cuatro secciones.

Ξ_{ij}**:** Ajuste entre osciladores.

β_o**:** Ajuste de capacidad de acoplamiento entre osciladores.

o_i, o_j**:** Posición de osciladores.

***hCrN*:** *hCr* normalizado respecto a la cantidad total de pixeles *K*.

σo_{zc}**:** Desfasamiento de radio de vecindario de Wz_c.

$Y_{8zc}(i,j)$**:** Salida de la GSCM con conectividad 8.

***A_blob*:** Áreas de las regiones de $Y_{8zc}(i,j)$.

r_d**:** índice de regiones en $Y_{8zc}(i,j)$.

μ_h**:** Medición de similitud entre $hI(q)$ y $hC_r(q)$.

$A\_Max(z_c)$**:** Máxima región de *A_blob*.

$ObjArea(z_c)$**:** Relación de áreas de regiones.

$STM(i,j,n_1)^{z_c}$**:** Memoria de corto plazo para PBSeg.

$V_4(x,y)^{z_c}$**:** Entrada a V4.

$V_{b4}(x,y)^{z_c}$**:** Segunda capa de V4.

$V_{4c}(x,y)^{z_c}$**:** Tercera capa de V4.

$V_{a4}(x,y)^{z_c}$**:** Primer capa de segmentación de V4.

$\boldsymbol{I_h(x,y)}$**:** Información de matiz agrupada en cuatro clases.

$\boldsymbol{SV_4(x,y)}$**:** Sumatoria de valores de $V_{b4}(x,y)^{z_c}$.

$\boldsymbol{Is_p(x,y)^t}$**:** Segmentación estática de PBSeg.

$\boldsymbol{Ig_p(x,y)^t}$**:** Gradiente de $Is_p(x,y)^t$.

$\boldsymbol{V_{f4}(x_a,y_a,w_a)}$**:** Arreglo que contiene secciones de $Ig_p(x,y)^t$.

$\boldsymbol{\Gamma_{f4}(x_a,y_a,w_a)}$**:** Arreglo que contiene secciones de $\Gamma_q(x,y)^t$.

$\boldsymbol{Fv_p(x_a,y_a,w_a)^t}$ **:** Arreglo que contiene secciones de $Ig_p(x,y)^t$ en frecuencia.

$\boldsymbol{F_{\Gamma q}(x_a,y_a,w_a)^t}$ **:** Arreglo que contiene secciones de $\Gamma_q(x,y)^t$ en frecuencia.

$\boldsymbol{\mu_f}$**:** Similitud entre $Fv_p(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$.

$\boldsymbol{\omega_a}$**:** índice de cuadro de las secciones de $Fv_p(x_a,y_a,w_a)$ y $F_{\Gamma q}(x_a,y_a,w_a)$.

$\boldsymbol{\mu^{z_c}_{v4}}$: Cantidad de pixeles que se mantienen activos en cada canal.

$\boldsymbol{RF\Theta_{z_c}}$**:** Reajuste de campos receptivos de los LGN.

$\boldsymbol{\varsigma s(RF\Theta_{z_c})}$**:** Reajuste de los LGN.

7. Red GSCM

$\boldsymbol{n_z}$**:** Iteraciones de la GSCM.

***NZ*:** Cantidad máxima de iteraciones de la GSCM.

$\boldsymbol{E_{zc}(i,j)^{n_z}}$**:** Función de umbral.

$\boldsymbol{\alpha_F z}$**:** Constante de retardo del alimentador.

$\boldsymbol{\alpha_L z}$**:** Constante de retardo del encadenador.

$\boldsymbol{U_{zc}(i,j)^{n_z}}$**:** Agregación de las sinapsis.

$\boldsymbol{Y_{zc}(i,j)^{n_z}}$**:** Salida de la PCNN.

$\boldsymbol{Wz_c}$**:** Pesos de entrada para el alimentador.

$\boldsymbol{\sigma_{zc}}$**:** Radio de vecindario de Wz_c.

8. Método de segmentación de objetos estáticos

$\boldsymbol{I_{bc}(x,y,z)^t}$**:** Salida de la T4CNN, donde $z=\{H,H_2,S,V\}$.

$\boldsymbol{i_L}$**:** Nivel de gris.

$\boldsymbol{h_{bc}(i_L,z)}$**:** Histogramas de $I_{bc}(x,y,z)^t$.

$\boldsymbol{r_c}$**:** Índice de clase.

$\boldsymbol{r_z(t)}$**,** $\boldsymbol{r_H}$**,** $\boldsymbol{r_S}$ ***y*** $\boldsymbol{r_V}$**:** Regiones de *kmeans*.

L_s: Resolución de niveles de grises.

LsMem: Memoria de imágenes.

***$I_{NS}(x,y,r_s)$*:** Diferencia entre moda de *LsMem* e $I_{sj}(x,y)^t$.

***$Is_e(x,y)^t$*:** Segmentación estática.

***$I_{sj}(x,y)^t$*:** Arreglo de posibles imágenes a segmentar.

ΔH: Intervalo de matiz.

***$Cr(r_c)^t$*:** Centros para el *kmeans*.

***$E_b(x,y,r_c)$*:** Distancia euclidiana entre centros y pixeles.

***$U(x,y,r_c)$*:** Región de KCNN.

***$\mu_{NS}(r_s)$*:** Valor esperado de $I_{Ns}(x,y,r_s)$.

m_{js}: Mínimo de $\mu_{NS}(r_s)$.

Secuencias de video

tp: Verdaderos positivos.

tn: Verdaderos negativos.

fp: Falsos positivos.

fn: Falsos negativos.

P: *Presicion.*

***Rc*:** *Recall.*

***F1*.** *F measure*

***Si*:** *Similarity.*

***FPR*:** Taza de falsos positives.

***FNR*:** Taza de falsos negativos.

***PWC*:** Porcentajes de pixeles clasificados erróneamente.

ANEXO

Resultados de métricas de base de datos de *www.changedetection.net* al dia 6 de abríl de 2014.

Tabla de resultados generales

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
STLBP [30]	6.14	0.7231	0.9993	0.0007	0.2769	2.9861	0.8067	0.9798
SGMM-SOD [28]	10	0.6396	0.9971	0.0029	0.3604	1.6846	0.7353	0.9471
Chebyshev probability approach [12]	10.29	0.694	0.9962	0.0038	0.306	1.3285	0.7259	0.891
Chebyshev prob. with Static Object detection [15]	10.43	0.6887	0.9963	0.0037	0.3113	1.4283	0.723	0.8906
Spectral-360 [26]	11.29	0.7238	0.9939	0.0061	0.2762	1.6337	0.7764	0.9114
PBAS [14]	11.71	0.7283	0.9934	0.0066	0.2717	1.5398	0.7556	0.8922
PBAS-PID [34]	11.86	0.7275	0.9936	0.0064	0.2725	1.5416	0.7539	0.8942
KDE - ElGammal [3]	12.14	0.6725	0.9955	0.0045	0.3275	1.6795	0.7423	0.8974
PSP-MRF [9]	14.29	0.5991	0.9962	0.0038	0.4009	1.9189	0.6932	0.9218
GPRMF [32]	14.57	0.8666	0.9917	0.0083	0.1334	1.9628	0.8305	0.8145
DPGMM [25]	15	0.8869	0.9882	0.0118	0.1131	1.5773	0.8134	0.7629
CDPS [24]	15.14	0.6195	0.995	0.005	0.3805	1.5205	0.6619	0.9014
CwisarD [29]	15.29	0.7357	0.9918	0.0082	0.2643	1.7664	0.7619	0.8007
Multi-Layer Background Subtraction [27]	15.86	0.5072	0.9986	0.0014	0.4928	3.8704	0.6331	0.9611
UBA [20]	16.14	0.688	0.9939	0.0061	0.312	1.6684	0.7283	0.7962
SC-SOBS [16]	16.86	0.6003	0.9957	0.0043	0.3997	1.9841	0.6923	0.8857
SGMM [21]	17	0.5363	0.997	0.003	0.4637	3.9394	0.6481	0.9263
Bayesian Background [17]	17.29	0.6026	0.9952	0.0048	0.3974	2.8676	0.6969	0.8877
GMM | KaewTraKulPong [2]	18	0.3395	0.9993	0.0007	0.6605	4.8419	0.4767	0.9709
Local-Self similarity [7]	18.29	0.9036	0.9692	0.0308	0.0964	3.2612	0.7297	0.6433
SOBS [1]	18.57	0.5888	0.9956	0.0044	0.4112	2.0983	0.6834	0.8754
GMM | RECTGAUSS-Tex [11]	18.86	0.2461	0.9994	0.0006	0.7539	5.2656	0.3682	0.9619
Histogram [23]	19.29	0.6412	0.9933	0.0067	0.3588	1.9669	0.6996	0.811
KNN [19]	19.43	0.4817	0.997	0.003	0.5183	4.3783	0.6046	0.9186
KDE - Integrated Spatio-temporal Features [8]	19.71	0.4147	0.9981	0.0019	0.5853	5.4152	0.4989	0.9164
Mahalanobis distance [5]	21.14	0.627	0.9906	0.0094	0.373	2.3462	0.7065	0.8617
TUBITAK UZAY 1 [33]	21.14	0.6589	0.992	0.008	0.3411	3.9821	0.6877	0.8127
KDE - Spatio-temporal change detection [10]	21.57	0.4065	0.9973	0.0027	0.5935	5.1527	0.5199	0.8761
GMM | Stauffer & Grimson [18]	21.71	0.5691	0.9946	0.0054	0.4309	4.2642	0.6621	0.8652
Quasi-Continuous Histograms based Motion Detection[22]	21.71	0.335	0.9982	0.0018	0.665	5.1493	0.4651	0.8784
GMM | Zivkovic [4]	22.57	0.5542	0.9942	0.0058	0.4458	4.3002	0.6548	0.8706
Euclidean distance [6]	24.71	0.5111	0.9907	0.0093	0.4889	3.8516	0.6313	0.8877

Tabla de resultados de categoría *baseline*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
SC-SOBS [16]	4.71	0.9327	0.998	0.002	0.0673	0.3747	0.9333	0.9341
SOBS [1]	7	0.9193	0.998	0.002	0.0807	0.4332	0.9251	0.9313
PSP-MRF [9]	7.14	0.9319	0.9978	0.0022	0.0681	0.4127	0.9289	0.9261
GPRMF [32]	7.43	0.906	0.9979	0.0021	0.094	0.4669	0.928	0.9524
Spectral-360 [26]	9.86	0.9615	0.9968	0.0032	0.0385	0.4263	0.933	0.9066
SGMM-SOD [28]	10.29	0.9334	0.9974	0.0026	0.0666	0.5494	0.9212	0.9113
PBAS-PID [34]	10.43	0.9585	0.9971	0.0029	0.0415	0.4822	0.9254	0.8971
DPGMM [25]	10.57	0.9632	0.9969	0.0031	0.0368	0.4949	0.9286	0.8984
Multi-Layer Background Subtraction [27]	10.71	0.8456	0.9984	0.0016	0.1544	0.8993	0.9004	0.9655
KDE - ElGammal [3]	11.57	0.8969	0.9977	0.0023	0.1031	0.5499	0.9092	0.9223
PBAS [14]	11.71	0.9594	0.997	0.003	0.0406	0.4858	0.9242	0.8941
CwisarD [29]	13.43	0.8989	0.9971	0.0029	0.1011	0.663	0.9075	0.9171
Histogram [23]	13.71	0.8777	0.9972	0.0028	0.1223	0.6679	0.9004	0.9254
CDPS [24]	13.86	0.9488	0.9965	0.0035	0.0512	0.6238	0.9208	0.8969
STLBP [30]	15	0.8	0.9988	0.0012	0.2	1.1411	0.8379	0.917
Bayesian Background [17]	15.14	0.7327	0.9984	0.0016	0.2673	0.9037	0.8271	0.962
KNN [19]	17	0.7934	0.9979	0.0021	0.2066	1.284	0.8411	0.9245
Mahalanobis distance [5]	17.43	0.8872	0.9963	0.0037	0.1128	0.729	0.8954	0.9071
Chebyshev prob. with Static Object detection [15]	17.43	0.8266	0.997	0.003	0.1734	0.8304	0.8646	0.9143
GMM | KaewTraKulPong [2]	18.29	0.5863	0.9987	0.0013	0.4137	1.9381	0.7119	0.9532
Local-Self similarity [7]	19	0.9732	0.9865	0.0135	0.0268	1.3352	0.8494	0.7564
GMM | Zivkovic [4]	19.29	0.8085	0.9972	0.0028	0.1915	1.3298	0.8382	0.8993
Euclidean distance [6]	19.43	0.8385	0.9955	0.0045	0.1615	1.026	0.872	0.9114
GMM | RECTGAUSS-Tex [11]	19.86	0.6669	0.9979	0.0021	0.3331	1.5342	0.75	0.9175
SGMM [21]	21.29	0.868	0.9949	0.0051	0.132	1.2436	0.8594	0.8584
UBA [20]	22	0.9017	0.9912	0.0088	0.0983	1.0169	0.8132	0.7423
GMM | Stauffer & Grimson [18]	24.29	0.818	0.9948	0.0052	0.182	1.5325	0.8245	0.8461
TUBITAK UZAY 1 [33]	25.57	0.8936	0.9847	0.0153	0.1064	2.0851	0.7633	0.6767
KDE - Integrated Spatio-temporal Features [8]	26.29	0.7472	0.9954	0.0046	0.2528	1.8058	0.7392	0.7998
KDE - Spatio-temporal change detection [10]	26.86	0.7551	0.994	0.006	0.2449	1.9154	0.7554	0.7833
Quasi-Continuous Histograms based Motion Detection[22]	29.43	0.7044	0.9923	0.0077	0.2956	2.2142	0.6616	0.7009

Tabla de resultados de categoría *camera Jitter*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
GPRMF [32]	2.14	0.8159	0.9953	0.0047	0.1841	1.1062	0.8596	0.9244
CwisarD [29]	5	0.7645	0.9916	0.0084	0.2355	1.7886	0.7814	0.8091
STLBP [30]	5.43	0.8797	0.9841	0.0159	0.1203	1.9191	0.8272	0.8124
DPGMM [25]	8.14	0.6988	0.993	0.007	0.3012	1.7707	0.7477	0.8426
PSP-MRF [9]	8.43	0.8211	0.9825	0.0175	0.1789	2.2781	0.7502	0.7009
Multi-Layer Background Subtraction [27]	11	0.6903	0.9905	0.0095	0.3097	2.1628	0.7311	0.7905
PBAS [14]	11.14	0.7373	0.9838	0.0162	0.2627	2.4882	0.722	0.7586
Spectral-360 [26]	11.14	0.6709	0.9906	0.0094	0.3291	2.0806	0.7156	0.8392
SGMM [21]	12	0.7088	0.9869	0.0131	0.2912	2.3761	0.7251	0.7752
KDE - Integrated Spatio-temporal Features [8]	12.43	0.7316	0.9857	0.0143	0.2684	2.4238	0.711	0.6993
PBAS-PID [34]	12.43	0.7278	0.9848	0.0152	0.2722	2.4939	0.722	0.7515
SGMM-SOD [28]	12.86	0.6113	0.9907	0.0093	0.3887	2.3608	0.6724	0.804
KDE - Spatio-temporal change detection [10]	13	0.7562	0.9816	0.0184	0.2438	2.745	0.7122	0.6793
SOBS [1]	13.14	0.8007	0.9787	0.0213	0.1993	2.7479	0.7086	0.6399
SC-SOBS [16]	13.86	0.8113	0.9768	0.0232	0.1887	2.8794	0.7051	0.6286
KNN [19]	15.43	0.7351	0.9778	0.0222	0.2649	3.1104	0.6894	0.7018
Bayesian Background [17]	17.43	0.5441	0.9886	0.0114	0.4559	2.8807	0.5988	0.6678
GMM | KaewTraKulPong [2]	17.57	0.5074	0.9888	0.0112	0.4926	3.0233	0.5761	0.6897
Chebyshev prob. with Static Object detection [15]	18.71	0.7223	0.9725	0.0275	0.2777	3.6203	0.6416	0.596
GMM | Stauffer & Grimson [18]	19	0.7334	0.9666	0.0334	0.2666	4.2269	0.5969	0.5126
TUBITAK UZAY 1 [33]	19.14	0.8646	0.9439	0.0561	0.1354	5.8825	0.5661	0.4247
KDE - ElGammal [3]	19.57	0.7375	0.9562	0.0438	0.2625	5.1349	0.572	0.4862
GMM | RECTGAUSS-Tex [11]	20.14	0.7649	0.9497	0.0503	0.2351	5.6663	0.537	0.4179
Local-Self similarity [7]	21.71	0.9764	0.6158	0.3842	0.0236	36.957	0.2074	0.1202
GMM | Zivkovic [4]	22.57	0.69	0.9665	0.0335	0.31	4.4057	0.567	0.4872
Color Histogram Backprojection [13]	22.57	0.4688	0.9821	0.0179	0.5312	3.7175	0.4822	0.5296
Mahalanobis distance [5]	23.14	0.7356	0.9431	0.0569	0.2644	6.439	0.496	0.3813
Euclidean distance [6]	24.43	0.7115	0.9456	0.0544	0.2885	6.2957	0.4874	0.3753
CDPS [24]	25	0.6025	0.9613	0.0387	0.3975	5.3593	0.4865	0.4397
Histogram [23]	26.43	0.7111	0.8412	0.1588	0.2889	16.2797	0.2784	0.1756

Tabla de resultados de categoría *dynamic Background*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
DMB [31]	3.29	0.9155	0.9987	0.0013	0.0845	0.2282	0.8262	0.7877
STLBP [30]	3.57	0.8559	0.9999	0.0001	0.1441	0.1508	0.9057	0.9922
CwisarD [29]	7.43	0.8355	0.9982	0.0018	0.1645	0.3389	0.8086	0.8096
DPGMM [25]	8	0.8852	0.9966	0.0034	0.1148	0.4121	0.8137	0.7762
Spectral-360 [26]	9.14	0.7748	0.9993	0.0007	0.2252	0.3464	0.7872	0.859
Chebyshev probability approach [12]	9.29	0.8182	0.9982	0.0018	0.1818	0.3436	0.7656	0.7633
Chebyshev prob. with Static Object detection [15]	10.29	0.8182	0.9976	0.0024	0.1818	0.4086	0.752	0.7339
PBAS-PID [34]	11.71	0.7662	0.9988	0.0012	0.2338	0.47	0.7352	0.8245
PBAS [14]	12.86	0.6955	0.9989	0.0011	0.3045	0.5394	0.6829	0.8326
GPRMF [32]	12.86	0.8991	0.9877	0.0123	0.1009	1.2694	0.7726	0.7414
SGMM-SOD [28]	14.29	0.7786	0.9966	0.0034	0.2214	0.6041	0.6883	0.7044
KDE - Spatio-temporal change detection [10]	14.57	0.8935	0.9908	0.0092	0.1065	1.0142	0.6574	0.5888
KNN [19]	14.86	0.8047	0.9937	0.0063	0.1953	0.8059	0.6865	0.6931
CDPS [24]	15	0.759	0.9947	0.0053	0.241	0.7281	0.7495	0.8086
GMM | KaewTraKulPong [2]	15.43	0.6303	0.9983	0.0017	0.3697	0.5405	0.6697	0.77
PSP-MRF [9]	15.43	0.8955	0.9859	0.0141	0.1045	1.4514	0.696	0.6576
Quasi-Continuous Histograms based Motion Detection[22]	17.57	0.8909	0.9896	0.0104	0.1091	1.1301	0.643	0.5347
TUBITAK UZAY 1 [33]	17.86	0.8176	0.992	0.008	0.1824	1.0428	0.6078	0.5903
SGMM [21]	18.29	0.7715	0.9933	0.0067	0.2285	0.9132	0.638	0.6665
KDE - Integrated Spatio-temporal Features [8]	18.71	0.8401	0.9908	0.0092	0.1599	1.1501	0.6016	0.5413
SC-SOBS [16]	18.71	0.8918	0.9836	0.0164	0.1082	1.6899	0.6686	0.6283
GMM | Stauffer & Grimson [18]	18.86	0.8344	0.9896	0.0104	0.1656	1.2083	0.633	0.5989
SOBS [1]	19.71	0.8798	0.9843	0.0157	0.1202	1.6367	0.6439	0.5856
GMM | Zivkovic [4]	20.43	0.8019	0.9903	0.0097	0.1981	1.1725	0.6328	0.6213
Multi-Layer Background Subtraction [27]	21.29	0.7584	0.9912	0.0088	0.2416	1.0758	0.6278	0.6466
Bayesian Background [17]	22.86	0.5962	0.9917	0.0083	0.4038	1.2427	0.5369	0.6898
KDE - ElGammal [3]	24.29	0.8012	0.9856	0.0144	0.1988	1.6393	0.5961	0.5732
Local-Self similarity [7]	24.43	0.8983	0.7694	0.2306	0.1017	22.7868	0.0949	0.0518
Mahalanobis distance [5]	25.86	0.8132	0.9698	0.0302	0.1868	3.1407	0.5261	0.4517
Euclidean distance [6]	27.43	0.7757	0.9714	0.0286	0.2243	3.0095	0.5081	0.4487
Histogram [23]	27.57	0.8069	0.9401	0.0599	0.1931	6.0488	0.2426	0.1516
GMM | RECTGAUSS-Tex [11]	28	0.4776	0.9838	0.0162	0.5224	1.9735	0.4296	0.6478
Color Histogram Backprojection [13]	31.14	0.6307	0.8906	0.1094	0.3693	11.0493	0.2675	0.198

Tabla de resultados de categoría *Intermittent Object Motion*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
SGMM-SOD [28]	3.57	0.7363	0.9909	0.0091	0.2637	2.5238	0.7151	0.8141
CDPS [24]	5.71	0.8084	0.9765	0.0235	0.1916	3.465	0.7406	0.7624
UBA [20]	6.29	0.7205	0.9827	0.0173	0.2795	3.0544	0.6886	0.731
PBAS-PID [34]	10	0.6997	0.9762	0.0238	0.3003	4.0991	0.6261	0.7041
KDE - Integrated Spatio-temporal Features [8]	10.29	0.4512	0.9964	0.0036	0.5488	4.4191	0.5454	0.8166
PBAS [14]	11.14	0.67	0.9751	0.0249	0.33	4.2871	0.5745	0.7045
Spectral-360 [26]	11.43	0.5945	0.9811	0.0189	0.4055	5.4443	0.5656	0.7192
SC-SOBS [16]	12.14	0.7237	0.9613	0.0387	0.2763	5.2207	0.5918	0.5896
SGMM [21]	12.86	0.5013	0.9853	0.0147	0.4987	4.918	0.5397	0.6993
KDE - Spatio-temporal change detection [10]	13.14	0.4372	0.9923	0.0077	0.5628	4.6997	0.5039	0.7212
GMM | Stauffer & Grimson [18]	13.57	0.5142	0.9835	0.0165	0.4858	5.1955	0.5207	0.6688
KNN [19]	14.14	0.4617	0.9865	0.0135	0.5383	5.137	0.5026	0.7121
PSP-MRF [9]	15.57	0.701	0.953	0.047	0.299	6.0594	0.5645	0.5727
SOBS [1]	16.29	0.7057	0.9507	0.0493	0.2943	6.1324	0.5628	0.5531
GMM | Zivkovic [4]	16.29	0.5467	0.9712	0.0288	0.4533	5.4986	0.5325	0.6458
DPGMM [25]	16.86	0.6763	0.947	0.053	0.3237	6.8457	0.5418	0.6525
CwisarD [29]	16.86	0.7847	0.9003	0.0997	0.2153	10.0314	0.5674	0.5013
GMM | RECTGAUSS-Tex [11]	17.86	0.219	0.9977	0.0023	0.781	5.2547	0.3146	0.585
GMM | KaewTraKulPong [2]	18	0.3476	0.9892	0.0108	0.6524	5.9854	0.3903	0.6953
TUBITAK UZAY 1 [33]	18.57	0.5823	0.9645	0.0355	0.4177	6.3265	0.5091	0.5533
Chebyshev prob. with Static Object detection [15]	19	0.357	0.9807	0.0193	0.643	6.47	0.3863	0.7688
STLBP [30]	19.29	0.2569	0.9923	0.0077	0.7431	5.9051	0.3129	0.5722
Local-Self similarity [7]	20	0.9027	0.8222	0.1778	0.0973	15.8827	0.5329	0.4445
Histogram [23]	20.29	0.7512	0.8656	0.1344	0.2488	13.1359	0.5112	0.4859
Quasi-Continuous Histograms based Motion Detection[22]	20.57	0.4407	0.9797	0.0203	0.5593	6.049	0.4367	0.5384
Multi-Layer Background Subtraction [27]	20.71	0.5012	0.9629	0.0371	0.4988	7.0245	0.4816	0.6024
Mahalanobis distance [5]	21.57	0.7165	0.8886	0.1114	0.2835	11.5341	0.4968	0.4535
Euclidean distance [6]	21.71	0.5919	0.9336	0.0664	0.4081	8.9975	0.4892	0.4995
GPRMF [32]	22.29	0.6101	0.8753	0.1247	0.3899	13.0733	0.487	0.5863
KDE - ElGammal [3]	24.57	0.5035	0.9309	0.0691	0.4965	10.0695	0.4088	0.4609
Bayesian Background [17]	25.43	0.4813	0.9304	0.0696	0.5187	9.9632	0.4081	0.4747

Tabla de resultados de categoría *Shadow*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
GPRMF [32]	2.57	0.9253	0.9922	0.0078	0.0747	1.0712	0.8889	0.8671
Spectral-360 [26]	4.57	0.9366	0.9905	0.0095	0.0634	1.1784	0.8843	0.8412
PBAS-PID [34]	6.57	0.9142	0.9906	0.0094	0.0858	1.2627	0.8617	0.8171
SGMM-SOD [28]	6.71	0.9191	0.9902	0.0098	0.0809	1.2534	0.8646	0.8226
PBAS [14]	7.86	0.9133	0.9904	0.0096	0.0867	1.2753	0.8597	0.8143
STLBP [30]	8.29	0.8125	0.9944	0.0056	0.1875	1.49	0.8416	0.89
DPGMM [25]	10.29	0.8545	0.9916	0.0084	0.1455	1.5947	0.8127	0.824
CwisarD [29]	10.29	0.8872	0.9897	0.0103	0.1128	1.3757	0.8412	0.8056
Multi-Layer Background Subtraction [27]	10.86	0.8588	0.9912	0.0088	0.1412	1.5621	0.8216	0.8099
Chebyshev probability approach [12]	11.71	0.8669	0.9887	0.0113	0.1331	1.5552	0.8333	0.8104
Chebyshev prob. with Static Object detection [15]	12.43	0.867	0.9887	0.0113	0.133	1.5561	0.8333	0.8103
KDE - Integrated Spatio-temporal Features [8]	14.14	0.7197	0.993	0.007	0.2803	2.1292	0.7545	0.8244
CDPS [24]	14.14	0.9233	0.9846	0.0154	0.0767	1.9516	0.8092	0.7567
SGMM [21]	15	0.858	0.9889	0.0111	0.142	1.7965	0.7944	0.7617
KNN [19]	16.14	0.7478	0.9916	0.0084	0.2522	2.0569	0.7468	0.7788
GMM | KaewTraKulPong [2]	16.57	0.6323	0.9936	0.0064	0.3677	2.3015	0.7176	0.8577
KDE - ElGammal [3]	16.57	0.8536	0.9885	0.0115	0.1464	1.6881	0.8028	0.766
PSP-MRF [9]	18.29	0.8736	0.9829	0.0171	0.1264	2.2414	0.7907	0.7281
Bayesian Background [17]	20.43	0.6537	0.9916	0.0084	0.3463	2.4695	0.6955	0.7791
SC-SOBS [16]	20.86	0.8502	0.9834	0.0166	0.1498	2.3	0.7786	0.723
SOBS [1]	21.43	0.835	0.9836	0.0164	0.165	2.3366	0.7716	0.7219
GMM | RECTGAUSS-Tex [11]	21.86	0.7189	0.9886	0.0114	0.2811	2.4111	0.7331	0.784
KDE - Spatio-temporal change detection [10]	22	0.697	0.9898	0.0102	0.303	2.4865	0.7136	0.7559
GMM | Stauffer & Grimson [18]	22	0.796	0.9871	0.0129	0.204	2.1951	0.737	0.7156
TUBITAK UZAY 1 [33]	22.14	0.8594	0.9768	0.0232	0.1406	2.8893	0.7509	0.6843
GMM | Zivkovic [4]	22.29	0.777	0.9878	0.0122	0.223	2.1957	0.7319	0.7232
UBA [20]	22.86	0.9084	0.9707	0.0293	0.0916	3.225	0.7123	0.6095
Local-Self similarity [7]	23.14	0.9584	0.9442	0.0558	0.0416	5.5496	0.5951	0.4673
Quasi-Continuous Histograms based Motion Detection[22]	24	0.6949	0.9887	0.0113	0.3051	2.587	0.7072	0.7378
Euclidean distance [6]	26.14	0.8001	0.9783	0.0217	0.1999	2.8987	0.6785	0.6112
Histogram [23]	27.43	0.8308	0.9686	0.0314	0.1692	3.7098	0.6589	0.6009
Mahalanobis distance [5]	28.43	0.7845	0.9708	0.0292	0.2155	3.7896	0.6348	0.5685

Tabla de resultados de categoría *Thermal*.

Method	Average ranking	Average Re	Average Sp	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision
STLBP [30]	6.14	0.7231	0.9993	0.0007	0.2769	2.9861	0.8067	0.9798
SGMM-SOD [28]	10	0.6396	0.9971	0.0029	0.3604	1.6846	0.7353	0.9471
Chebyshev probability approach [12]	10.29	0.694	0.9962	0.0038	0.306	1.3285	0.7259	0.891
Chebyshev prob. with Static Object detection [15]	10.43	0.6887	0.9963	0.0037	0.3113	1.4283	0.723	0.8906
Spectral-360 [26]	11.29	0.7238	0.9939	0.0061	0.2762	1.6337	0.7764	0.9114
PBAS [14]	11.71	0.7283	0.9934	0.0066	0.2717	1.5398	0.7556	0.8922
PBAS-PID [34]	11.86	0.7275	0.9936	0.0064	0.2725	1.5416	0.7539	0.8942
KDE - ElGammal [3]	12.14	0.6725	0.9955	0.0045	0.3275	1.6795	0.7423	0.8974
PSP-MRF [9]	14.29	0.5991	0.9962	0.0038	0.4009	1.9189	0.6932	0.9218
GPRMF [32]	14.57	0.8666	0.9917	0.0083	0.1334	1.9628	0.8305	0.8145
DPGMM [25]	15	0.8869	0.9882	0.0118	0.1131	1.5773	0.8134	0.7629
CDPS [24]	15.14	0.6195	0.995	0.005	0.3805	1.5205	0.6619	0.9014
CwisarD [29]	15.29	0.7357	0.9918	0.0082	0.2643	1.7664	0.7619	0.8007
Multi-Layer Background Subtraction [27]	15.86	0.5072	0.9986	0.0014	0.4928	3.8704	0.6331	0.9611
UBA [20]	16.14	0.688	0.9939	0.0061	0.312	1.6684	0.7283	0.7962
SC-SOBS [16]	16.86	0.6003	0.9957	0.0043	0.3997	1.9841	0.6923	0.8857
SGMM [21]	17	0.5363	0.997	0.003	0.4637	3.9394	0.6481	0.9263
Bayesian Background [17]	17.29	0.6026	0.9952	0.0048	0.3974	2.8676	0.6969	0.8877
GMM | KaewTraKulPong [2]	18	0.3395	0.9993	0.0007	0.6605	4.8419	0.4767	0.9709
Local-Self similarity [7]	18.29	0.9036	0.9692	0.0308	0.0964	3.2612	0.7297	0.6433
SOBS [1]	18.57	0.5888	0.9956	0.0044	0.4112	2.0983	0.6834	0.8754
GMM | RECTGAUSS-Tex [11]	18.86	0.2461	0.9994	0.0006	0.7539	5.2656	0.3682	0.9619
Histogram [23]	19.29	0.6412	0.9933	0.0067	0.3588	1.9669	0.6996	0.811
KNN [19]	19.43	0.4817	0.997	0.003	0.5183	4.3783	0.6046	0.9186
KDE - Integrated Spatio-temporal Features [8]	19.71	0.4147	0.9981	0.0019	0.5853	5.4152	0.4989	0.9164
Mahalanobis distance [5]	21.14	0.627	0.9906	0.0094	0.373	2.3462	0.7065	0.8617
TUBITAK UZAY 1 [33]	21.14	0.6589	0.992	0.008	0.3411	3.9821	0.6877	0.8127
KDE - Spatio-temporal change detection [10]	21.57	0.4065	0.9973	0.0027	0.5935	5.1527	0.5199	0.8761
GMM | Stauffer & Grimson [18]	21.71	0.5691	0.9946	0.0054	0.4309	4.2642	0.6621	0.8652
Quasi-Continuous Histograms based Motion Detection[22]	21.71	0.335	0.9982	0.0018	0.665	5.1493	0.4651	0.8784
GMM | Zivkovic [4]	22.57	0.5542	0.9942	0.0058	0.4458	4.3002	0.6548	0.8706
Euclidean distance [6]	24.71	0.5111	0.9907	0.0093	0.4889	3.8516	0.6313	0.8877