EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Estudio de Algoritmos de IA Aplicables al Estegoanálisis de Imágenes Digitales

presentada por

Ing. Sergio Alexis Ramírez Valle

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Raúl Pinto Elías

Codirector de tesis

Dr. Dante Mújica Vargas

Cuernavaca, Morelos, México. Febrero de 2021.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 29/enero/2021

OFICIO No. DCC/040/2021
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

C. DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **C. Ing. Sergio Alexis Ramírez Valle**, con número de control M18CE073, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"Estudio de algoritmos de IA aplicables al estegoanálisis de imágenes digitales"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

Dr. Rafal Pinto Elías
Doctor en Ciencias en la Especialidad de Ingeniería Eléctrica
3890453
Director de tesis

Dr. Dante Mújica Vargas
Doctor en Comunicaciones y Electrónica
09131756
Co-director de tesis

Dra. Andrea Magadán Salazar
Doctorado en Ciencias Computacionales
10654097
Revisor 1

Dr. Jorge Alberto Fuentes Pacheco
Doctor en Ciencias de la Computación
8533951
Revisor 2

C.c.p. Depto. Servicios Escolares
Expediente / Estudiante
JGGS/lmz

Cuernavaca, Mor., 03/febrero/2021
No. de Oficio: SAC/37/2021
Asunto: Autorización de impresión de tesis

SERGIO ALEXIS RAMÍREZ VALLE
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado ***"Estudio de algoritmos de IA aplicables al estegoanálisis de imágenes digitales",*** ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
"Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO SUBDIRECCIÓN ACADÉMICA

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C.c.p. M.E. Guadalupe Garrido Rivera. Jefa del Departamento de Servicios Escolares
Expediente
CMAZ/CHG

Dedicatorias

A Dios por permitirme seguir adelante y mostrarme el camino que debo seguir, por darme fuerza y determinación.

A mis padres que me han forjado desde un inicio para ser una mejor persona y estar preparado para resolver los retos de la vida.

A mis hermanos que han demostrado ser unas personas distinguidas con fortaleza.

A toda mi familia a la que aprecio, admiro y respeto mucho.

Agradecimientos

Agradezco el apoyo económico otorgado por CONACyT, en el programa de becas para estudios de maestría y al TecNM/CENIDET por facilitar sus instalaciones.

Agradezco al TecNM/CENIDET por permitirme realizar la maestría.

Un total agradecimiento a mis directores: al Dr. Raúl Pinto Elías por las deducciones, dirección y consejos hacia éste trabajo. Al Dr. Dante Mújica Vargas por su gran apoyo, tiempo y dedicación.

A mis revisores: a la Dra. Andrea Magadán Salazar y al Dr. Jorge Alberto Fuentes Pacheco, por los valiosos aportes y consejos, su tiempo y esfuerzos, siempre mostraron un gran empeño.

A mis profesores que hicieron todo lo posible por la enseñanza de los grandes conocimientos.

Al grupo de la especialidad de Inteligencia Artificial, por las grandes enseñanzas y situaciones memorables.

Al mis compañeros y amigos de la generación 2018ca, también a mis amigos de la cueva, por las grandes convivencias y apoyo.

A todas aquellas personas que contribuyeron de forma directa o indirecta en la realización de este trabajo.

Resumen

La esteganografía y el estegoanálisis son temas importantes en la ocultación y la seguridad de la información. La esteganografía tiene como objetivo ocultar información en objetos sin levantar sospechas. El estegoanálisis es la contramedida de la esteganografía, se encarga de detectar los objetos que han sido inyectados.

En este trabajo se presenta un método para generar una sub-imagen con el objetivo de maximizar el nivel del mensaje inyectado. Después de obtener la sub-imagen se aplica la extracción de características de alta dimensión. Finalmente, se entrena un clasificador binario para crear un modelo y poder distinguir entre dos clases: las imágenes limpias e inyectadas. Se seleccionaron dos de los algoritmos más frecuentes de esteganografía reportados en el estado del arte. El método propuesto fue probado con el algoritmo HILL y WOW con diferentes *payloads*. Los resultados muestran que el método propuesto tiene un rendimiento cercano en comparación el enfoque SRM de la literatura.

Abstract

Steganography and steganalysis are important topics in hiding and information security. Steganography aims to hide information in objects without raising suspicions. Steganalysis is the countermeasure of steganography, it is in charge of detecting the objects that have been embedded.

In this work a method to generate a sub-image in order to maximize the level of the embedded message is proposed. After obtaining the sub-image, the extraction of high-dimensional features is applied. Finally, a binary classifier is trained to build a model to distinguish two classes: cover and stego images. We selected two of the most frequent steganography algorithms reported in the state of the art. The proposed method was tested with the HILL and WOW algorithm with different payloads. The results show that the proposed method has close performance compared to the SRM literature approach.

Contenido

Índice de figuras

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Anexos

Índice de Tablas

Capítulo 2

Capítulo 5

Capítulo 6

Anexos

Acrónimos

Acrónimo	Significado
ABC	Artificial Bee Colony
A-MSPU	Adaptive More Surrounding Pixels Using
bpac	Bits per nonzero AC DCT coefficient
bpp	Bits per pixel
CDF	Cross-Domain Feature
CMD	Clustering Modification Directions
CSM	Cover source mismatch
DCT	Discrete Cosine Transform
EA	Edge-Adaptive
EMD	Exploiting modification direction
FLD	Fisher Linear Discriminant
GLCM	Gray level Co-occurance Matrix
GLM	Gray Level Modification
GUI	Graphical User Interface
HILL	High-pass, Low-pass and Low-pass
HUGO	Highly Undetectable steGO
IA	Inteligencia Artificial
IDE	Integrated Development Environment
IFAB	Image steganalysis based on Feature selection using Artificial Bee colony
KB	Ker-Bohme
LSB	Least Significant Bit
LSBM	Least Significant Bit Matching
MBNS	Multi-Base Notation System
MiPOD	Minimizing the Power of Optimal Detector
nsF5	no-shrinkage F5

Acrónimo	Significado
OOB	Out-Of-Bag
POV	Pairs of Values
PPM	Pixel Pair Matching
RAW	Formato bruto
RGB	Red, Green, Blue
RISAB	Region based Image Steganalysis using Artificial Bee colony
RQR	Raw Quick Pair
SPAM	Subtractive Pixel Adjacency Model
SRM	Spatial Rich Model
SSIS	Spread Spectrum Image Steganography
STC	Syndrome-Trellis Codes
Sub2SRM	Enfoque propuesto, Sub-imagen más SRM más clasificador ensamblado. Se prueba con imágenes de 512x512
SubSRM	Mismo que SubSRM, pero se prueba con sub-imágenes de 384x384
SVM	Support Vector Machine
UNIWARD	Universal wavelet relative distortion
S-UNIWARD	Spatial-Universal wavelet relative distortion
WOW	Wavelet Obtained Weights
XOR	Exclusive Or

Capítulo 1

Introducción y descripción del problema

1.1 Introducción

La esteganografía es un método para transmitir información desde un emisor hacia un receptor mediante un canal público, con la particularidad de que la información que se quiere transferir está oculta en un objeto portador. De manera análoga a la criptografía, la esteganografía permite que se intercambien mensajes en secreto, pero a diferencia de la criptografía, la esteganografía agrega otra capa de seguridad al ocultar la acción de intercambio de información entre ambas partes. Es decir, la información que se quiere trasmitir no debe de levantar sospecha de que hay un canal para intercambiar información.

El principal objetivo de la esteganografía es ocultar la presencia de la información que se envía en un canal de comunicación. Para ocultar información se necesita de un objeto que se comunique entre el emisor y el receptor, éste objeto no está definido a un cierto tipo, por ejemplo, en la arquitectura cliente-servidor, específicamente en el protocolo de comunicación, un potencial objeto portador serían los paquetes de datos que se envían por medio de la red. Para establecer la comunicación oculta los canales pueden ser de cualquier tipo. Otros ejemplos de objetos portadores, las imágenes digitales, los videos, elementos de audio, texto, documentos. Se aplica lo mismo para los mensajes que se inyectan en los objetos portadores, es decir, pueden ser de distintos tipos.

La contramedida de la esteganografía se conoce como estegoanálisis que se encarga de detectar la trasmisión de mensajes por parte de la esteganografía en los objetos que se capturan y se analizan. Un objeto portador ideal, son las imágenes digitales, este tipo de objetos son muy adecuados debido a que son ampliamente utilizados y tienen redundancia de información. En esta investigación se utilizan en el proceso de inyección, análisis y detección.

La técnica más conocida de esteganografía en imágenes digitales es Sustitución del Bit Menos Significativo o LSB [Chan, 2004], esta técnica se trata de utilizar el bit menos significativo de un pixel e inyectar el mensaje en éste y los siguientes bits de los otros pixeles. La forma gráfica se presenta en la Figura 1.1.

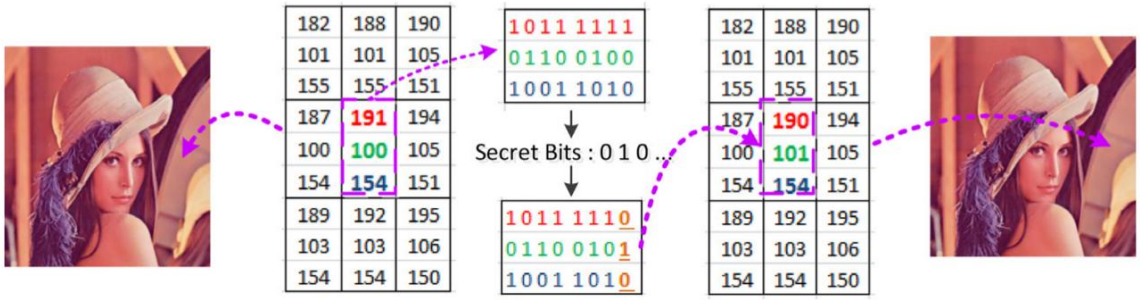


Figura 1.1. Sustitución del Bit Menos Significativo [Hussain, 2018].

Con los avances referentes a la ocultación de mensajes, se han presentado técnicas más complejas, las cuales son altamente indetectables estadísticamente. Debido a la complejidad de estas técnicas se han planteado nuevos enfoques para su detección por parte del estegoanálisis. Lo que conlleva que las técnicas de estegoanálisis generen características de alta dimensión [Holub, 2013] [Chen, 2013]. La rama de la esteganografía y estegoanálisis en muy amplia. Para trabajar en la detección, una vertiente sería establecer un balance entre el rendimiento de detección y la dimensión de características, como oportunidades de mejora.

1.2 Descripción del problema

1.2.1 Complejidad

La gran cantidad de técnicas de esteganografía y sus diferentes enfoques dificultan la contramedida, el estegoanálisis. Aún cuando se apliquen las técnicas de Inteligencia Artificial, no podrían diferenciar del ruido en una imagen. Es decir, que las técnicas que se utilicen puedan encontrar una diferencia entre una alteración “normal” en una imagen y una alteración causada por inyección de la esteganografía.

Existen técnicas en esteganografía que utilizan una codificación para incrustar el mensaje en forma adaptativa y no son fácilmente detectables por las técnicas de estegoanálisis. Otra circunstancia que causa problemas para el estegoanálisis, es cuando el mensaje es fragmentado e incrustado en diferentes imágenes u objetos portadores, es decir, la cantidad de bits por pixel (bpp) que se inyecta es sumamente reducida y por consecuencia, los métodos de contramedida no lograrían identificar cada objeto como corrupto o limpio [Ker, 2006].

Los ataques de estegoanálisis generalmente se quedan en la fase de detección binaria (si el objeto está limpio o no), y es poco probable que el ataque de estegoanálisis pueda recuperar al mensaje incrustado en la imagen. Y si el mensaje es recuperado, el problema crece cuando el mensaje que se transmite se le aplica una técnica de cifrado.

Otro escenario es cuando en estegoanálisis se entrena con un conjunto y se aplica para detectar un conjunto diferente, lo que ocasiona que el error de detección aumente debido a la falta de relación entre ambos conjuntos. En esteganografía, esta situación se reconoce como *Cover Source Mismatch* (CSM) [Kodovsky, 2014].

1.2.2 Análisis del problema

En la literatura se ha mostrado que una de las formas más eficaces para ocultar mensajes en objetos es en las imágenes digitales, dadas sus características. La esteganografía es un campo muy extenso de investigación, porque permite explotar áreas que en un inicio no se pensaron para transmitir información, como se enunció anteriormente emplea muchos canales para establecer comunicaciones ocultas. Esta investigación solo se encamina en este tipo de objetos digitales.

Los elementos que conforman un sistema de esteganografía en imágenes digitales son los siguientes, y se muestran en la Figura 1.2.

a) Técnica de incrustación. La función de incrustación o inyección es un algoritmo o mecanismo que se encarga de ocultar el mensaje en un objeto portador. Generalmente utilizan dos parámetros de entrada: el mensaje a ocultar y el objeto portador. La salida es un estego-objeto.
b) Técnica de extracción. Se conoce como función de recuperación a la parte del sistema de esteganografía que se encarga de recuperar el mensaje del objeto portador.
c) Parámetros. El parámetro compartido entre ambas funciones, es la *Clave Estego*. Para la función de inyección, se necesita una imagen y como último parámetro el mensaje que se desea enviar.

Este trabajo de investigación tiene por objetivo detectar con las técnicas de IA si hay inyección en una imagen digital. Para ello, se estudió, se implementó y se evaluó si las técnicas de IA pueden caracterizar o si pueden determinar qué imágenes han sido inyectadas con técnicas de esteganografía. Para esta tarea, por el lado de la esteganografía, se escogieron algoritmos del estado del arte, adaptativos al contenido de la imagen al inyectar el mensaje. Lo que dificulta la caracterización de las imágenes, llevando el problema a generar características de alta dimensión [Fridrich, 2012] [Ker ,2013], para establecer una separabilidad entre un objeto inyectado y uno limpio.

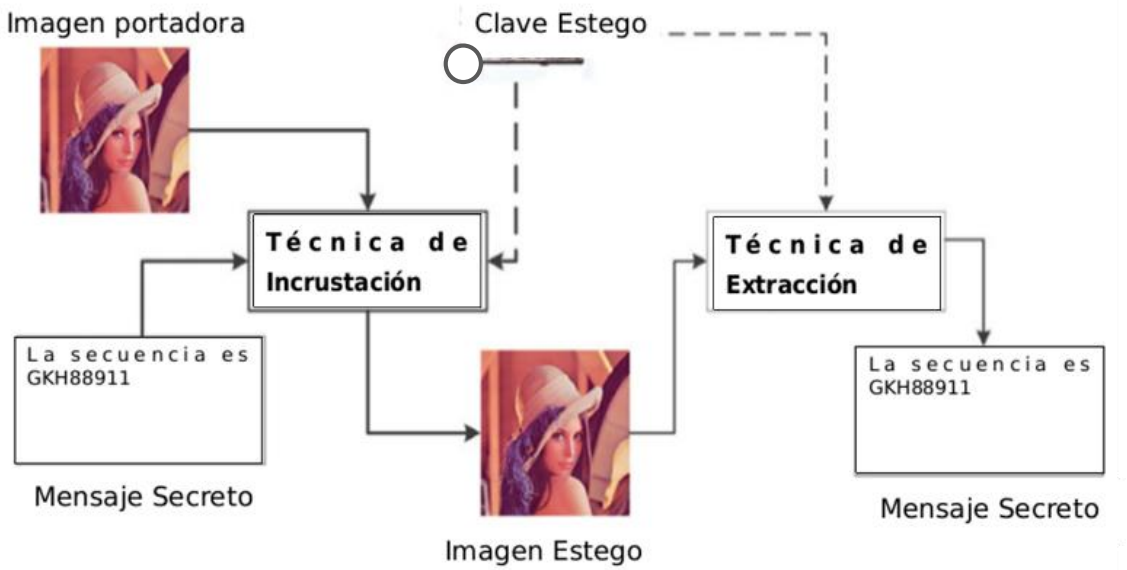


Figura 1.2. Sistema de esteganografía [Hussain, 2018].

1.3 Organización de la tesis

Esta tesis se expone de la siguiente manera:

- Capítulo 2. Presenta los fundamentos teóricos para comprender esta investigación. También, se realiza una revisión de técnicas de esteganografía y de su contramedida el estegoanálisis, reportadas en el estado el arte. Con base en esto se menciona de manera rápida la propuesta de solución.

- Capítulo 3. Se analiza el problema de forma específica, y se desarrolla la propuesta de solución.
- Capítulo 4. Describe el análisis y diseño del sistema, así como la arquitectura del mismo. Seguido de la implementación del sistema de esteganografía y estegoanálisis, así como una interfaz de usuario.
- Capítulo 5. Se muestran los diferentes experimentos realizados con los algoritmos de esteganografía seleccionados, desplegando imágenes inyectadas como ejemplos. Se caracterizan los conjuntos de imagen con los diferentes *payloads*. Después se realiza un análisis de los resultados obtenidos.
- Capítulo 6. Se evalúa el cumplimiento del objetivo y los alcances planteados en la propuesta de la tesis, expuestos en el capítulo 2, en base a las pruebas y resultados obtenidos.
- Anexos. Se presenta un ejemplo de codificación, se detalla un ejemplo del descriptor global empleado, se explica cómo se calculan los parámetros usados del clasificador en la etapa de entrenamiento y prueba, y se muestra el código para entrenamiento y clasificación.

Capítulo 2

Contexto teórico y estado del arte

2.1 Contexto teórico

2.1.1 Esteganografía

La esteganografía es la habilidad de transmitir información a través de objetos aparentemente inocentes de una manera que la presencia del mensaje es desconocida [Anderson, 1998]. El término esteganografía en griego significa literalmente, “escritura oculta".

La esteganografía es el arte de la comunicación indetectable en la que los mensajes se incrustan en objetos de aspecto inofensivo, como una imagen digital [Pevny, 2007].

2.1.2 Estegoanálisis

La contramedida de la esteganografía es el estegoanálisis que se interpreta como el arte y la ciencia de exponer los datos secretos ocultos por la esteganografía [Johnson, 1998].

El estegoanálisis, desde la perspectiva del oponente, es un arte de detener las comunicaciones encubiertas mientras evita afectar a los objetos inocentes. Su requisito básico es determinar con precisión si un mensaje secreto está oculto en el objeto de prueba. Otros requisitos más dedicados pueden incluir averiguar el tipo de esteganografía, estimar la longitud aproximada del mensaje, y por último extraer el mensaje oculto [Bhattacharyya, 2011].

2.1.3 El problema del prisionero

El estegoanálisis se refiere a las técnicas que ayudan a Wendy a distinguir entre los objetos de portadores y los de inyectados, como se muestra en la Figura 2.1. Wendy tiene que hacer esta distinción sin ningún conocimiento de la clave secreta que Alice y Bob pueden estar compartiendo y, a veces, incluso sin ningún conocimiento del algoritmo específico que podrían estar usando para incrustar el mensaje secreto [Simmons, 1984].

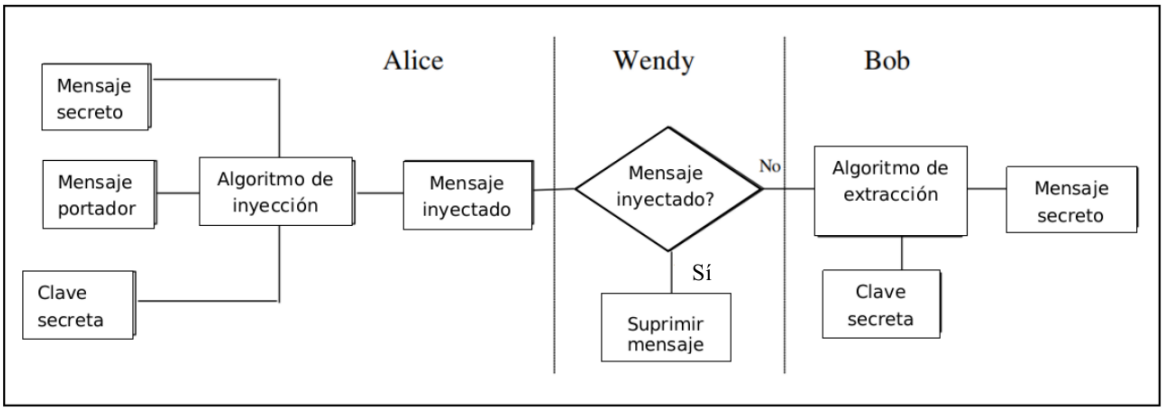


Figura 2.1. El problema del prisionero [Simmons, 1984].

2.1.4 Carga de inyección / *payload*

El *payload* es una característica que se encarga de medir la cantidad de datos que se pueden incrustar en una imagen portadora, se representa con *bpp* (bits por pixel) que es el número máximo de bits que se pueden ocultar. De acuerdo con Pradhan en [Pradhan, 2016] es la cantidad máxima de datos que se pueden ocultar en la imagen. Por ejemplo si la imagen es de 256x256. Si se quiere ocultar un mensaje de 75000 bits se divide entre la cantidad de pixeles de la imagen, es decir, 256x256 = 65536. Esto es igual a 1.144 bpp. Por lo tanto, bpp = (mensaje en bits/cantidad total de pixeles de la imagen).

2.1.5 Imagen portadora e inyectada

a) Imagen portadora. Es un objeto *cover* que hace referencia a que es de tipo imagen que no contiene ningún mensaje, es un objeto normal sin alteración por la esteganografía.

b) Imagen inyectada. Es un objeto *stego* que fue modificado por la esteganografía, es decir que contiene un mensaje o parte de un mensaje.

2.2 Estado del arte

2.2.1 Técnicas de esteganografía

2.2.1.1 Técnicas clásicas

Un método fácil es el cambio del Bit Menos Significativo [Chan, 2004], algunas técnicas incluyen la sustitución LSB adaptativa basada en bordes, textura, nivel de intensidad y el brillo de las imágenes para la incrustación de datos [Yang, 2009]. Para mejorar la calidad visual en una imagen inyectada, en el trabajo de [Wu, 2003] se propuso una técnica de esteganografía basada en el valor de la diferencia de pixeles. El valor de la diferencia entre dos pixeles vecinos se utiliza para decidir cuántos bits secretos deben incrustarse. Este método incorpora una mayor capacidad del mensaje cuando las imágenes tienen una mayor imperceptibilidad visual, como por ejemplo, una imagen con ruido. En la Figura 2.2 se ilustra el método.

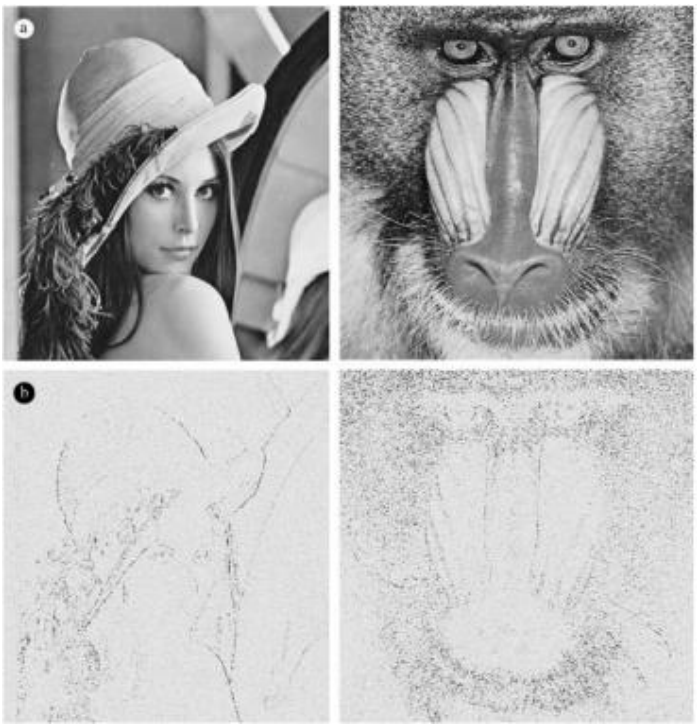


Figura 2.2. Imágenes portadoras (a), diferencias entre la imagen original y la inyectada (b) [Wu, 2003].

Otra técnica utiliza una base específica para determinar la variación local de la intensidad de pixeles en la imagen, cuando hay más zonas con textura la inyección del mensaje es mayor. En [Zhang, 2006], se muestra el uso de la Explotación de Modificación de Dirección (EMD, *Exploiting modification direction*), el dígito secreto fue trasformado por el sistema (*2n+1*), *n* es el número de pixeles de la imagen portadora. Otros métodos se basan en el Sistema de Notación Multi-base (MBNS, *Multi-Base Notation System*), el mecanismo que utilizaron fue reexpresar o trasformar los datos secretos en un sistema de notación antes del proceso de inyección. En las técnicas basadas en MBNS, los datos secretos se convirtieron en símbolos y se reexpresaron en el sistema de notación de base múltiple, es decir, en sistemas de números binarios, decimales y octales. Además, estos símbolos se incrustaron en las intensidades de los pixeles. Afrakhteh en [Afrakhteh, 2010], propusieron una adaptación de más pixeles circundantes utilizando el método Usando la Adaptación de Más Pixeles Circundantes (A-MSPU, *Adaptive More Surrounding Pixels Using*), que puede mejorar la imperceptibilidad visual de los problemas MBNS.

Un método interesante se basó en la combinación de Emparejamiento de Pares de Pixeles (PPM, *Pixel Pair Matching*) [Hong, 2012]. La forma general de PPM es usar los valores del par de pixeles como una coordenada de referencia y después buscar una coordenada en el conjunto de vecindad de este par de pixeles de acuerdo con el dígito del mensaje. El par de pixeles se reemplaza por la coordenada buscada para ocultar el dígito. Se expresa como (*pi,1 , pi,2*) como coordenada de referencia para buscar otra coordenada (*p'i,1 , p'i,2*) dentro de un conjunto de vecindad predefinido, el procedimiento se ilustra en la Figura 2.3. Técnicas más evidentes en cuanto a su funcionamiento, son aquellas que emplean la Modificación del Nivel de Gris (GLM, *Gray Level Modification*). Estos tipos de métodos de incrustación asignaron los datos secretos modificando el nivel de gris de los pixeles. En el trabajo de Potdar en [Potdar, 2004], se basaron en una función matemática, donde se seleccionó un grupo de pixeles para transformar los datos secretos. Cuando se identifican los bordes en una imagen se pueden incrustar datos en los pixeles, Luo en [Luo, 2010] realizó una incrustación adaptativa.

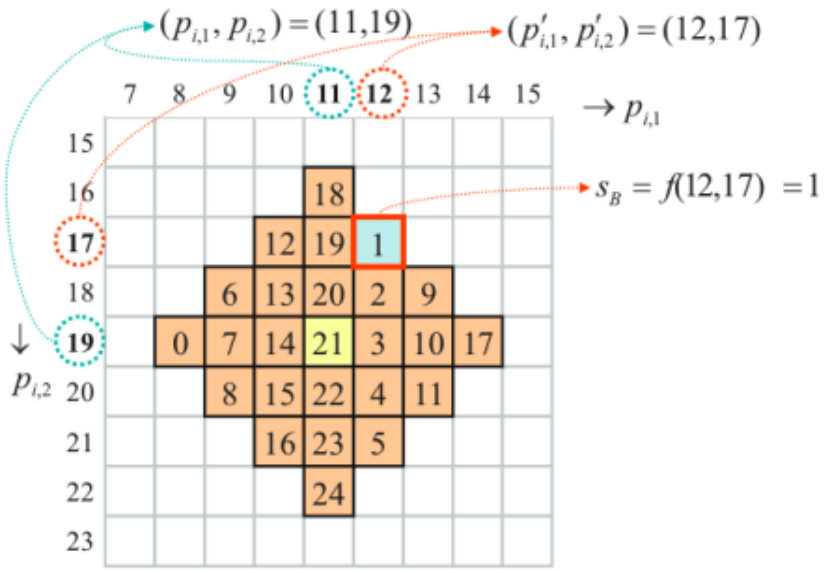


Figura 2.3. El vecindario de (11, 19), y sus correspondientes coordenadas [Hong, 2012].

Otras técnicas se basaron en trasformar la imagen, en el trabajo de Wang en [Wang, 2006], dividieron la imagen portadora y la imagen secreta en bloques de dos vías para buscar el bloque de mayor similitud en ambas imágenes. El bloque de dos vías consiste en dividir la imagen portadora y la imagen inyectada en varios bloques (sub-imágenes), después se comparan de dos bloques por cada imagen. Lo siguiente fue crear índices de los bloques emparejados y no emparejados para inyectarlos en los bits menos significativos de la imagen portadora. Existen métodos que se basan en un pixel indicador, en estos se emplean indicadores de pixel o bloque para integrar datos.

Gutub en [Gutub, 2010] propuso un método basado en indicadores de pixeles que tienen una carga útil alta. El método propuesto incorporó los datos secretos en uno o ambos canales de datos de una manera cíclica en el modelo de color RGB. En la Figura 2.4 se muestra el diagrama de flujo del método.

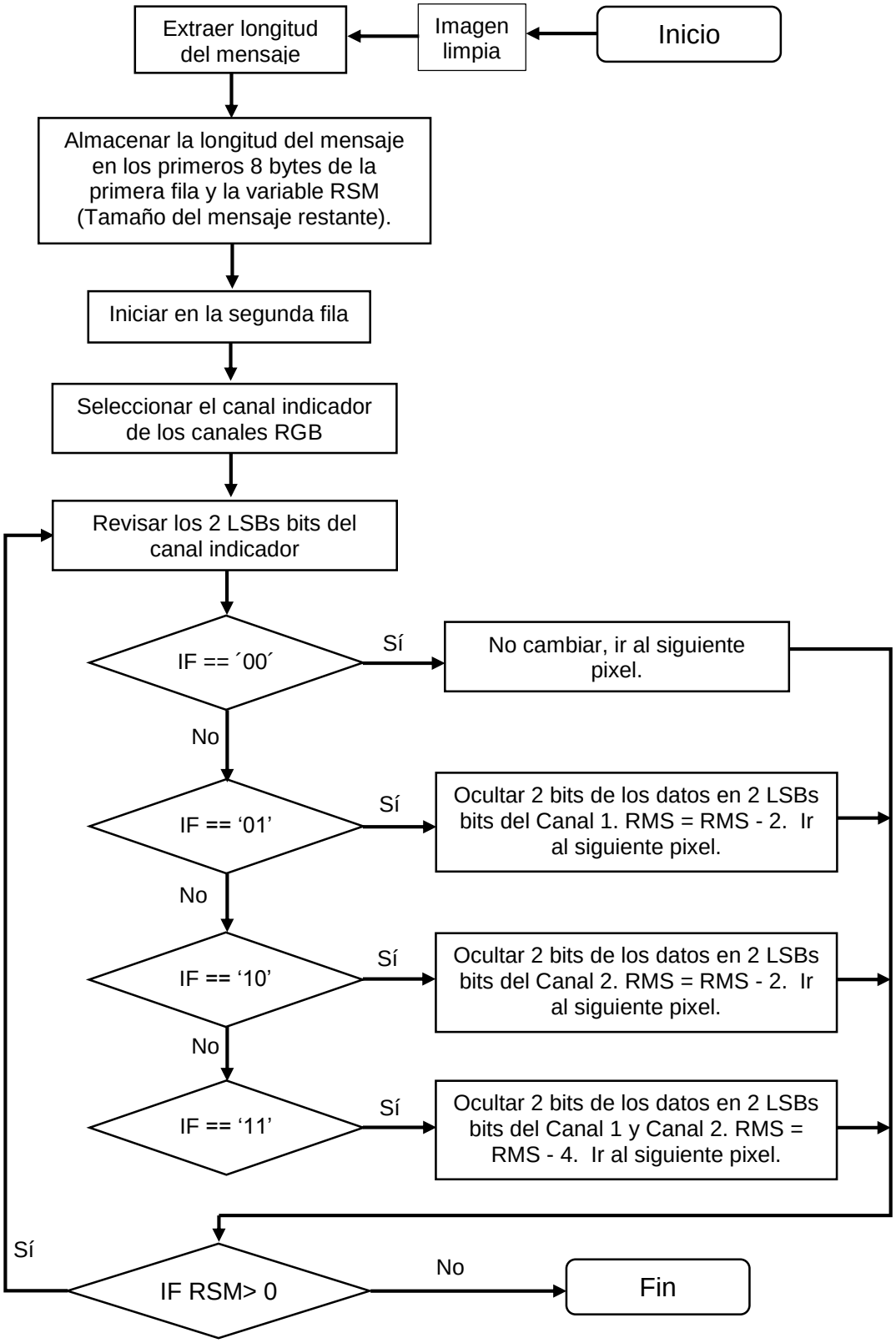


Figura 2.4. Proceso de ocultación [Gutub, 2010].

2.2.1.2 Técnicas adaptativas al contenido

Este enfoque de esteganografía en imágenes digitales utiliza un *framework* de codificación para minimizar una distorsión de inyección [Filler & Fridrich 2010] [Filler, 2010] [Filler, 2011]. Estos métodos son más difíciles de detectar que las técnicas de esteganografía clásicas basadas en sustituir el Bit Menos Significativo (LSB, *Least Significant Bit*) [Chan, 2004] [Hussain, 2018].

Algunos códigos (*framework*) están disponibles para experimentación, dejando como objetivo para los investigadores de esteganografía centrarse en crear una función de costo para minimizar la distorsión al generar una imagen inyectada con un mensaje. Comúnmente se usa STC (STC, *Syndrome-Trellis Codes*).

2.2.1.2.1 *Framework Syndrome-Trellis Codes*

El problema de la esteganografía adaptativa consiste en minimizar el impacto de inyección/incrustación al generar una imagen inyectada. El problema ya ha sido descrito conceptualmente por Crandall [Crandall ,1998]. Sugirió que siempre que el codificador inyecta como máximo un bit por pixel (bpp), debe hacer uso del impacto de incrustación definido por cada pixel y minimizar su suma total:

"Conceptualmente, el codificador examina un área de la imagen y asigna pesos a cada una de las opciones (pixeles) que permiten incrustar los bits deseados en esa área. Califica cada opción (puntuación) por lo ideal que es y elige la opción con la mejor puntuación".

Bierbrauer en [Bierbrauer, 1998] estudió un caso especial de este problema y describió una conexión entre los códigos y el problema de minimizar el número de pixeles modificados. Esta conexión, que se ha conocido como inyección matricial (codificación), se hizo famosa por Westfeld en [Westfeld, 2001] quien la incorporó en su algoritmo de esteganografía F5. Donde se utilizó el código de Hamming binario para implementar la codificación. Después, en la literatura se han propuesto diferentes opciones de codificación para este problema [Van, 2001] [Schönfeld, 2006] [Fridrich ,2006] [Bierbrauer, 2008] [Zhang, 2009].

Como una mejora de los trabajos anteriores se presentó el algoritmo de codificación STC (*Syndrome-Trellis Codes*). Se considera que $\boldsymbol{x} = (x_1, \dots, x_n), \boldsymbol{y} = (y_1, \dots, y_n) \in \{0,1\}^n$ como dos vectores binarios de una imagen limpia y su correspondiente inyectada de dimensión *n*, y el mensaje ***m*** como otro vector binario $\boldsymbol{m} \in \{0,1\}^m$. La ecuación (2.1) es una métrica de distorsión, que representa el impacto total de inyección en una imagen:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \boldsymbol{\rho_i} |\boldsymbol{x_i} - \boldsymbol{y_i}| \qquad (2.1)$$

donde:

$\boldsymbol{\rho_i}$ son los costos de cambiar de $\boldsymbol{x_i}$ a $\boldsymbol{y_i}$.

En la ecuación (2.1), cuando el costo en $\boldsymbol{\rho_i}$ se multiplica por el valor absoluto de $\boldsymbol{x_i} - \boldsymbol{y_i}$, cuando el valor de $\boldsymbol{y_i}$ no cambia, el resultado de la distorsión es cero. Dados *m-bits* de un mensaje se buscó que se inyectaran en *n-elementos* de un objeto portador, mientras se mantenía una distorsión (2.1) al mínimo posible. En la codificación *Syndrome* [MacKay, 2003] capítulo 25 y 48, la función de inyección y de extracción se realizan usando un código lineal binario *C* de longitud *n* y de longitud o rango *n-m*:

$$Emb(\boldsymbol{x}, \boldsymbol{m}) = \arg \min_{\boldsymbol{y} \in C(\boldsymbol{m})} D(\boldsymbol{x}, \boldsymbol{y}), \qquad (2.2)$$

$$Ext(\boldsymbol{y}) = \mathbb{H}\boldsymbol{y} \qquad (2.3)$$

Donde $\mathbb{H}\ \{0,1\}^{m \times n}$ es una matriz de comprobación de paridad del código C.

El algoritmo funciona de la siguiente manera. Primero, dado un mensaje ***m*** y una imagen ***x***, se selecciona una matriz $\mathbb{H}$ en una forma especial que permita representar cada solución de $\mathbb{H}\boldsymbol{y}$ = ***m*** como un camino a través de un enrejado (*Trellis*). La ***y*** optima más cercana a ***x*** se encuentra con el algoritmo Viterbi. La construcción de la matriz $\mathbb{H}$, se obtiene al colocar una pequeña matriz $\hat{\mathbb{H}}$ de un tamaño *h x w* como se muestra en la Figura 2.5. Las sub-matrices $\hat{\mathbb{H}}$ se colocan una al lado de la otra y se desplazan hacia abajo una fila. La altura *h* de la sub-

matriz (llamada *altura de restricción*) es un parámetro de diseño que afecta la velocidad y la eficiencia del algoritmo (típicamente, 6 ≤ h ≤ 15), por default está en 10. La matriz $\mathbb{H}$ es de tamaño [*αn*] × *n,* donde *α* es el *playload*. La submatriz $\hat{\mathbb{H}}$ actúa como un parámetro de entrada compartido entre el emisor y el receptor.

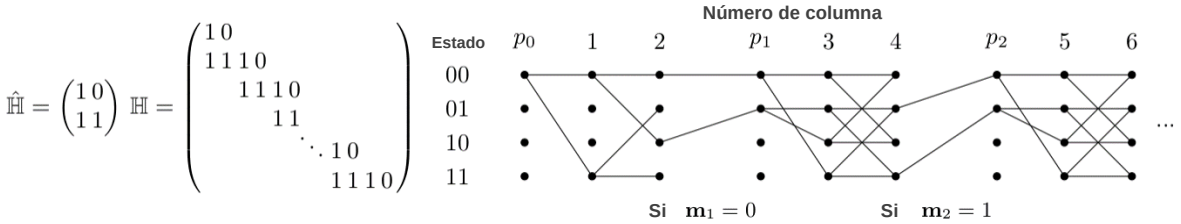


Figura 2.5. Ejemplo de una matriz $\mathbb{H}$ formada de una submatriz $\hat{\mathbb{H}}$ $(h = 2, w = 2)$ y su correspondiente *syndrome trellis.* El *syndrome trellis* consiste en bloques repetidos de *w+1* columnas, donde "p_0" y "p_i", i>0, denotan las columnas y poda, respectivamente. La columna denominada *l* ∈ {1, 2, ...} corresponde a la l-ésima columna de la matriz $\mathbb{H}$ [Filler ,2010].

El proceso de codificación se puede explicar mejor utilizando un enrejado de síndrome (*syndrome trellis o simplmenete trellis*) de la matriz de verificación de paridad que se muestra en la Figura 2.5. Cada $\boldsymbol{y}$ que satisface $\mathbb{H}\boldsymbol{y} = \mathbf{m}$ se representa como un camino a través del enrejado. Cada camino comienza en el extremo izquierdo del estado todo cero (00) en el enrejado y se extiende hacia la derecha.

De cada nodo se despliegan dos caminos, el primero es cuando no se realiza ninguna operación y el segundo cuando la operación XOR se ejecuta con el valor de la columna de $\mathbb{H}$ y el estado actual, y se agregan etiquetas. En ambos casos se le asignan los pesos $\boldsymbol{\rho_i}$, hasta ir avanzando y terminar de leer la matriz, al llegar al lado derecho del enrejado.

El problema es encontrar la ruta más corta, esto se puede resolver de forma eficiente con el algoritmo de Viterbi. Consiste en dos partes, hacia adelante (*forward*) y hacia atrás (*backward*). La parte hacia adelante, consiste en ir creando las posibles rutas y se realiza la poda cuando a un nodo le llega más de una ruta, es decir, se elige la más corta. Finaliza cuando se recorre todo el enrejado y se tiene la ruta más corta. En el paso hacia atrás, simplemente se recorre la ruta más corta

leyendo las etiquetas para formar el objeto ***y***. Un ejemplo del algoritmo se presenta en el Anexo A.

2.2.1.2.2 Función de Costo

Para fines prácticos, el *framework* que se describió en la sección anterior, recibe una función de costo y un nivel de carga (por ejemplo: 0.2 bpp, implica que se inyecta un mensaje en el 20% de la imagen), el mensaje es aleatorio. Lo que permite a los investigadores centrarse en diseñar una buena función de costo para la esteganografía.

Designing steganographic distortion using directional filters [Holub, 2012]: En este trabajo se diseñó un método que los autores llamaron Pesos Obtenidos de la Wavelet (WOW, *Wavelet Obtained Weights*). Todos los diseños de esteganografía más seguros para las imágenes digitales utilizan funciones de distorsión definidas heurísticamente que restringen los cambios de incrustación en aquellas partes de la imagen que son difíciles de modelar (por ejemplo, texturas complejas o áreas "ruidosas").

Una forma de definir la función de distorsión en el dominio espacial es asignar costos de pixel midiendo el impacto de cambiar cada pixel en un espacio de característica (modelo) utilizando una norma ponderada. Hacer que los pesos dependan de la vecindad local del pixel introduce adaptabilidad de contenido deseable.

Los autores emplearon un banco de filtros de paso alto direccionales para obtener los llamados residuos direccionales, que están relacionados con la previsibilidad del pixel en una dirección determinada. Al medir el impacto de la incrustación en cada residuo direccional y al agregar adecuadamente estos impactos, se forzó que el costo de incrustación sea alto cuando el contenido es predecible en al menos una dirección (áreas lisas y a lo largo de los bordes) y bajo

donde el contenido es impredecible en todas las direcciones (por ejemplo, en áreas con textura o ruidosas). De este modo, el algoritmo resultante se vuelve altamente adaptable y resiste mejor el estegoanálisis utilizando el Modelo Enriquecido Espacial (SRM, *Spatial Rich Model*) [Fridrich, 2012].

Filtros direccionales

El algoritmo debe incrustar el mensaje en áreas de textura/ruidosas que no fueran fácilmente modelables en ninguna dirección. Para este fin, evaluaron la suavidad en múltiples direcciones usando un banco de filtros $\boldsymbol{B_n = \{K^{(1)},\ldots,K^{(n)}\}}$ que consiste en *n* filtros de paso alto multidireccionales representados por sus kernels normalizados. Los filtros se muestran en la Figura 2.6. Los residuos con índice *k,* $\boldsymbol{R^{(k)}, k = 1, \ldots, n,}$ se calculan como $\boldsymbol{R^{(k)} = K^{(k)} \star X}$, donde " $\star$ " es la operación de convolución entre un filtro y la imagen limpia.

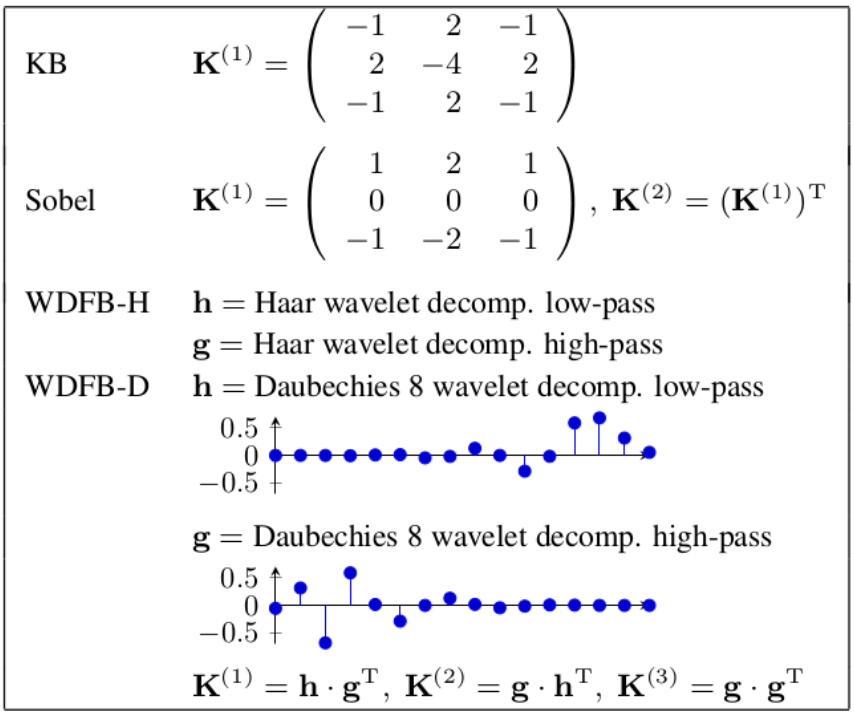


Figura 2.6. Banco de filtros [Holub, 2012].

Cálculo de la inyección ideal

La incrustación cambia valores grandes de residuos direccionales, donde están las texturas y los bordes, y preserva los valores pequeños, donde el contenido es predecible. Se calcula con la ecuación:

$$\xi_{ij}^{(k)} = \left|\boldsymbol{R}^{(k)}\right| \star \left|\boldsymbol{R}^{(k)} - \boldsymbol{R}_{[ij]}^{(j)}\right| \overset{(a)}{=} \left|\boldsymbol{R}^{(k)}\right| \star \left|\boldsymbol{K}^{(k)}\right|. \quad (2.4)$$

Cálculo de los costos de inyección

Para calcular los costos se utilizó la siguiente ecuación (2.5), los autores obtuvieron mejores resultados con el parámetro *p = -1*.

$$\rho_{ij}^{(p)} = \left(\sum_{k=1}^{n} \left|\xi_{ij}^{(k)}\right| p\right)^{-\frac{1}{p}} \quad (2.5)$$

A new cost function for spatial image steganography [Li, 2014]: En este artículo se propuso una nueva función de costo, asegurando que todos los pixeles dentro de las regiones de textura tuvieran costos relativamente bajos. La nueva función de costo que se propuso se diseñó mediante el uso de un filtro de paso alto y dos filtros de paso bajo, lo que hace que más cambios de incrustación se concentraran en las áreas de textura.

Lo símbolos $\boldsymbol{X} = (x_{i,j})^{n_2 \times n_1}, \boldsymbol{Y} = (y_{i,j})^{n_2 \times n_1}$ son usados para denotar a una imagen limpia de 8-bits en escala de grises y su versión inyectada. Los autores restringieron su diseño para el caso de incrustación ternaria, donde $y_{i,j} \in I_{i,j} = \{\min(x_{i,j} - 1, 0), x_{i,j}, \max(x_{i,j} + 1, 255)\}$. Usaron $|\boldsymbol{H}|$ para denotar la matriz que contiene los recíprocos de cada elemento de ***H.*** La notación $\otimes$ y $\odot$ representan la convolución *mirror-padded* y la correlación *mirror-padded* respectivamente.

Los autores propusieron una función de costo mejorada con los siguientes pasos:

1. Calcularon la incrustación ideal usando primero un filtro paso alto $\boldsymbol{H_1}$ en la imagen limpia para obtener los residuos del filtro, y luego usaron un filtro paso bajo $\boldsymbol{L_1}$ para los valores absolutos de los residuos resultantes (2.6).

$$\boldsymbol{\zeta^{(k)} = \left| X \otimes H^{(k)} \right| \otimes L_1} \quad (2.6)$$

2. Calcularon el valor del costo utilizando un filtro paso bajo $\boldsymbol{L_2}$ para la agregación de los valores recíprocos de las incrustaciones ideales (2.7).

$$\boldsymbol{\varrho = \left(\sum_{k=1}^{n} \frac{1}{\zeta^{(k)}} \right) \otimes L_2} \quad (2.7)$$

Mencionaron que se puede interpretar que usa un filtro de paso alto y un filtro paso bajo para localizar las regiones menos predecibles. Un único filtro de paso alto no direccional también puede lograr el objetivo de encontrar las áreas menos predecibles. Por lo tanto, explicaron que el proceso de obtención de valor de costo puede simplificarse aún más como (2.8).

$$\boldsymbol{\varrho = \frac{1}{|X \otimes H^{(1)}| \otimes L_1} \otimes L_2} \quad (2.8)$$

Selección de filtros

Hay tres filtros que utilizaron en la función de costo propuesta y afectaron el rendimiento general. Dado que el filtro 3 × 3 KB (Ker-Bohme) [Holub, 2012] tiene un rendimiento superior en estegoanálisis, lo seleccionaron como el filtro de paso alto en (2.8). Tiene la siguiente forma:

$$\boldsymbol{H^{(1)}} = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}. \quad (2.9)$$

Con la función de costo que propusieron, obtuvieron un mejor rendimiento con el filtro *KB*. Utilizaron el filtro promedio para $\boldsymbol{L_1}$ y $\boldsymbol{L_2}$ debido a su rápida implementación. El tamaño por defecto que reportaron fue de 3 × 3 para $\boldsymbol{L_1}$ y 15 × 15 para $\boldsymbol{L_2}$.

Universal distortion design for steganography in an arbitrary domain [Holub, 2014]: En este trabajo se utilizó la función de distorsión que tiene similitud con la de WOW (*Wavelet Obtained Weights*) (2.5) pero es más simple y adecuada para incrustar en un dominio arbitrario. Dado que la distorsión es en forma de una suma de cambios relativos entre las imágenes inyectadas y portadoras representadas en el dominio de wavelet, de ahí su nombre de distorsión relativa de wavelet universal (UNIWARD).

Se seleccionaron tres filtros $B = \{\boldsymbol{K}^{(1)}, \boldsymbol{K}^{(2)}, \boldsymbol{K}^{(3)}\}$, los residuos direccionales se obtienen $\boldsymbol{W}^{(k)} = \boldsymbol{K}^{(k)} \star \boldsymbol{X}$, donde se representa la convolución entre la imagen $\boldsymbol{X}$ y un kernel $\boldsymbol{K}$.

Dadas un par de imágenes $\boldsymbol{X}$ y $\boldsymbol{Y}$, representadas en el dominio espacial como, $\boldsymbol{W_{uv}^{(k)}}(\boldsymbol{X})$ y $\boldsymbol{W_{uv}^{(k)}}(\boldsymbol{Y}), k = 1,2,3, u \in \{1, \dots, n_1\}, v \in \{1, \dots, n_2\}$. La función de distorsión UNIWARD es la suma de los cambios relativos de todos los coeficientes de la wavelet con respecto a la imagen limpia (2.10):

$$D(X,Y) \triangleq \sum_{k=1}^{3} \sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \frac{\left| W_{uv}^{(k)}(\boldsymbol{X}) - W_{uv}^{(k)}(\boldsymbol{Y}) \right|}{\sigma + \left| W_{uv}^{(k)}(\boldsymbol{X}) \right|} \qquad (2.10)$$

donde $\sigma > 0$, es una constante para estabilizar las operaciones numéricas, en el artículo se estableció con $\sigma = 10$.

A strategy of clustering modification directions in spatial image steganography [Li, 2015]: En este trabajo los autores utilizaron los impactos de incrustación para mejorar el rendimiento de la indetectabilidad estadística para la esteganografía. La principal contribución es que desarrollaron una nueva estrategia, denominada Direcciones de Modificación de Agrupamiento (CMD, *Clustering Modification Directions*), que mejoró la seguridad de la esteganografía al diseñar funciones de costos heurísticas.

Los autores tomaron como referencia para su método una función ya diseñada como las anteriores descritas (2.5), (2.8), (2.10) y dividieron la imagen en partes iguales (sub-imágenes) para dividir el mensaje e ir actualizando los costos. La Figura 2.7 ilustra las sub-imágenes.

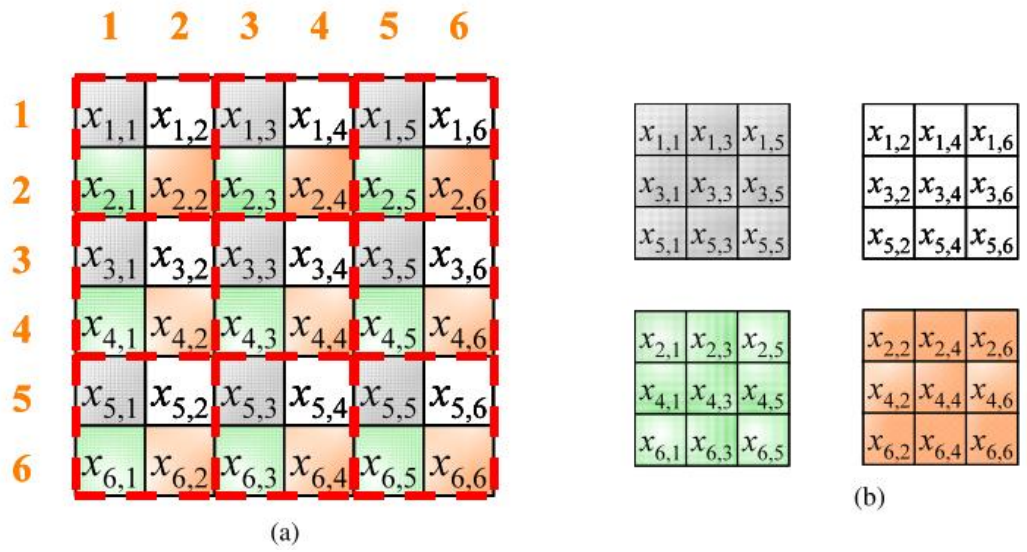


Figura 2.7. Un ejemplo de la división de una imagen en cuatro sub-imágenes. En un bloque de 2 x 2 de la imagen de ejemplo, los cuatro elementos son asignados a diferentes sub-imágenes. (a) Imagen de 6 x 6 pixeles y (b) cuatro Sub-imágenes [Li, 2015].

Presentaron un algoritmo práctico que asigna costos de acuerdo con dicha estrategia y es fácilmente aplicable a los esquemas aditivos propuestos. Específicamente, transformaron una imagen limpia en varias sub-imágenes. La distorsión dentro de una sub-imagen se define en una forma aditiva de modo que

los códigos de esteganografía prácticos existentes como en [Filler, 2011] se pueden emplear directamente.

Content-adaptive steganography by minimizing statistical detectability [Sedighi, 2015]: En este artículo fue propuesto un algoritmo de esteganografía que fue llamado MiPOD (*Minimizing the Power of Optimal Detector*) presentaron un enfoque alternativo basado en un estimador de variancia entre un bloque de pixeles. Este estimador se genera mediante dos pasos. El primer paso es suprimir el contenido de la imagen usando un filtro para quitar el ruido. El segundo paso, ajustaron el modelo paramétrico local a los vecinos de cada valor residual para obtener la estimación de la varianza final.

Cuando se conoce la varianza de cada pixel, se calcularon las tasas de cambio. Una vez que se calcularon las tasas de cambio, fue necesario convertirlas en costos para que la incrustación del mensaje real se pueda ejecutar con el marco bien establecido de códigos de STC. En la Figura 2.8 se ilustra el método general.

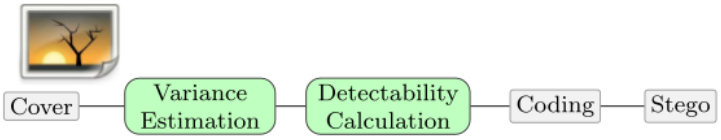


Figura 2.8. Método MiPOD [Sedighi, 2015].

Además, introdujeron una propuesta denominada "emisor limitado por la detección" (detectability-limited sender) que ajusta el tamaño de la carga útil de cada imagen para que no exceda un nivel prescrito de detección estadística dentro del modelo elegido.

2.2.2 Técnicas de estegoanálisis

2.2.2.1 Técnicas clásicas

Westfeld en [Westfeld, 1999] presentó la primera técnica de estegoanálisis estadístico. Utilizaron los Pares de Valores (POV, *Pairs of Values*). Los pares de valores pueden ser valores de pixeles, coeficientes de la Transformada de Coseno Discreta (DCT, *Discrete Cosine Transform*) o índices de paleta. Afirmaron que las frecuencias de cada uno de dos pares de pixeles en cada POV tienden a estar lejos de la media del POV. El ataque Chi-cuadrado detectó estos POV casi iguales en imágenes y, por consiguiente, en información incrustada. El método de Chi-cuadrada detectó de manera confiable los mensajes incrustados secuencialmente, pero tiene poco éxito cuando la incrustación es aleatoria. En la Figura 2.9 se muestra el método aplicado.

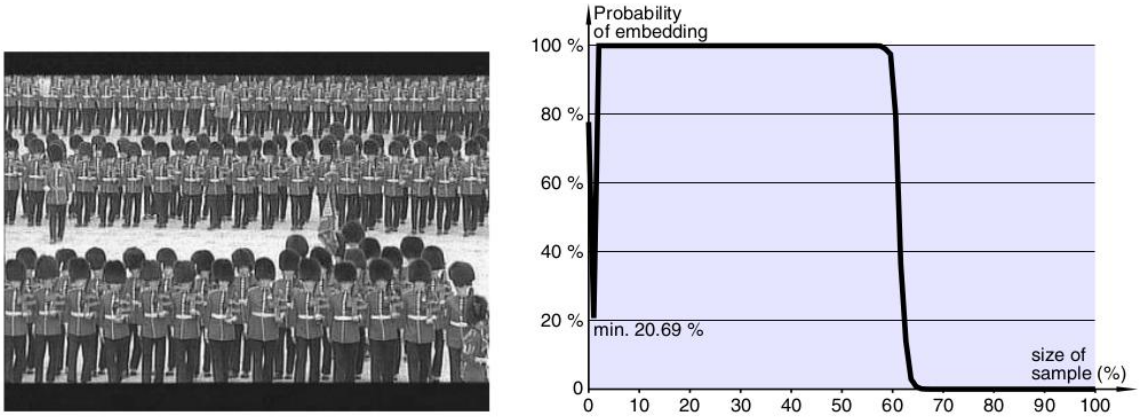


Figura 2.9. Herramienta Jsteg; con el 50% de inyección y las probabilidades de que tenga un mensaje oculto al aplicar el método [Westfeld, 1999].

Long en [Long, 2000] propuso un método para detectar la incrustación de LSB en imágenes de color de 24 bits, lo llamó Par Rápido en Bruto (RQR, *Raw Quick Pair*), tomó los pares de colores cercanos y los analizó, si los pares son difieren entonces fueron modificados por el LSB. El proceso de inyectar mensajes en imágenes aumenta el número de pares de colores cercanos. Por lo tanto, al contar el número de pares de colores cercanos se caracterizó una imagen como inyectada o limpia. Mostraron que incluso para capacidades de mensajes secretos

de 0.1 a 0.3 bits por pixel, es posible lograr un alto grado de fiabilidad en la detección. Desafortunadamente el método sólo se puede aplicar a imágenes a color.

En [Fridrich, 2001] se propuso otra técnica que funciona para imágenes en escala de grises y a color. La imagen analizada se dividió en bloques de pixeles y a cada uno se le midió el nivel de ruido. Después se hizo el cambio de LSB en una porción fija de pixeles en cada grupo y se determinó como regular o singular de acuerdo a si disminuyó o aumentó el nivel de ruido en cada uno de ellos. Este método resultó ser más confiable que el método Chi-cuadrada.

Para atacar la Esteganografía de Imagen de Espectro Expandido (SSIS, *Spread Spectrum Image Steganography*), que incrusta la información dentro del ruido y luego la inyecta a la imagen portadora. Li en [Li, 2013] desarrollaron un algoritmo de *kernel* de mínimos cuadrados multiportador iterativo de baja complejidad para extraer mensajes desconocidos, ocultos en *hosts* de imágenes a través de inyecciones de espectro expandido.

Pevny en [Pevny, 2010] formuló un método que llamó Matriz de Adyacencia de Pixel Sustractivo (SPAM, *Steganalysis by Subtractive Pixel Adjacency Matrix*), en este enfoque las dependencias entre los pixeles vecinos de la imagen filtrada (residuos de ruido) se modelaron como una cadena de Markov de orden superior. La matriz de probabilidad de transición de entrenamiento se usó como una característica vectorial para crear un modelo usando algoritmos de aprendizaje automático. Los experimentos realizados mostraron que el conjunto de características de SPAM también puede detectar de manera confiable los algoritmos que se ocultan en el dominio de la Transformada de Coseno Discreta (DCT, *Discrete Cosine Transform*). En la Figura 2.10, se muestra el proceso de extracción de características SPAM.

Sajedi en [Sajedi, 2016] propuso un enfoque que se constituye de dos etapas. En la primera, se analizó una base de imágenes para descubrir el patrón o la firma de las imágenes inyectadas. Se refirió a patrón al extraer las características notables de las imágenes inyectadas y sus valores relativos. Este patrón se construyó en

forma de un conjunto de reglas difusas *if – then*, que representaron la similitud entre las imágenes inyectadas. En la segunda etapa, el clasificador Máquina de vectores de soporte (SVM Support Vector Machine) fue entrenado para detectar un solo método de esteganografía a la vez. Después de descubrir los patrones del método de esteganografía usado de la imagen inyectada, se utilizó el modelo adecuado para analizarlo, el modelo contiene el patrón de cada algoritmo de inyección.

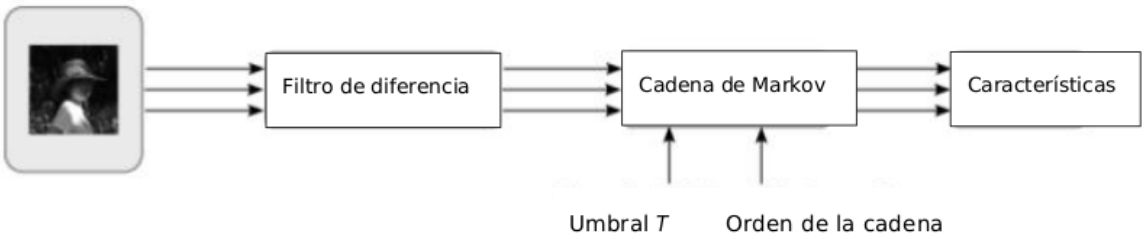


Figura 2.10. Diagrama de extracción de características SPAM [Pevny, 2010].

2.2.2.1 Técnicas de contramedida a la esteganografía adaptativa

Steganalysis of content-adaptive steganography in spatial domain [Fridrich, 2011]: En este trabajo se presentó una técnica para detectar al algoritmo de esteanografía adaptable al contenido, Altamente Indetectable SteGO (HUGO, *Highly Undetectable steGO*) [Pevny & Filler, 2010] e identificaron características capaces de detectar la carga útil incorporada utilizando espacios de características de alta dimensión para capturar las relaciones más sutiles entre los pixeles individuales. La dimensión del vector de característica fue de 33,963 elementos.

Posteriormente, utilizaron el clasificador ensamblado [Kodovsky, 2011] con la característica que llamaron *HOLMES,* y lo compararon con los resultados de la característica CDF (Cross-Domain Feature) [Kodovsky, 2010] que entrenaron con la SVM con un kernel Gaussiano. La característica CDF genera un vector de 1,234 elementos. Como banco de imágenes usaron BOSSbase versión 0.92 [Filler & Pevny, 2010] que contiene 9074 imágenes en escala de grises, en formato bruto (RAW) de tamaño 512x512. Para el entrenamiento utilizaron 8074 y para prueba

1000. Como métrica usaron el error de detección P_E. Los resultados se muestran en la Tabla 2.1.

Tabla 2.1. Resultados del modelo "H" con Clasificador ensamblado y CDF con *Gaussian*-SVM.

Payload (bpp)	H	CDF (G-SVM)
0.1	41.4	46.1
0.2	30.9	39.6
0.3	22.2	33.9
0.4	16.1	28.3
0.5	12.0	24.1

La métrica utilizada indica que si se tiene un valor cercano a 0 la clasificación es muy eficiente, mientras que si tiene un valor cercano a 100 indica que las clases no son separables. Los resultados de la Tabla 2.1 muestran que la característica H, tiene un mejor rendimiento para todos los niveles de *playload* en comparación con la característica CDF. Incluso cuando el *payload* fue más alto con 0.5 (bpp), lo cual implica un mayor número de pixeles modificados, la característica H tiene un poco más del doble de desempeño que CDF**.**

Region based image steganalysis using artificial bee colony [Mohammadi, 2017]: En este artículo se presentó un nuevo enfoque de análisis de imágenes mediante Colonias de Abejas Artificiales (ABC, *Artificial Bee Colony*), en el que se utilizó la selección de características y la extracción de características, mediante el Estegoanálisis de Imágenes Basada en una Región Usando Colonias de Abejas Artificiales (RISAB, *Region based Image Steganalysis using Artificial Bee colony*).

Su objetivo fue descubrir una sub-imagen que permitiera aumentar la precisión de la detección de imágenes inyectadas a partir de imágenes portadoras. Para este enfoque se utilizó la técnica Estegoanálisis de Imágenes basado en la Selección de Características utilizando Colonia de Abejas artificiales (IFAB*, Image steganalysis based on Feature selection using Artificial Bee colony*) para seleccionar características, que fue propuesto en [Mohammadi, 2014].

El método que se propuso incluye dos fases separadas: la fase de entrenamiento y fase de prueba. En la fase de entrenamiento implementaron la Colonia de abejas artificiales (ABC) aplicando la densidad como función de aptitud para detectar la mejor subimagen con longitud (*L*) y ancho (*W*), y luego lo verificaron y validaron utilizando el descriptor de textura conocido como energía.

Finalmente, se creó un modelo de aprendizaje basado en energía y características seleccionadas con IFAB. Sin embargo, la cantidad de imágenes utilizadas para sus experimentos fue tan solo de 900. En la Figura 2.11, se expone la obtención de una sub-imagen.

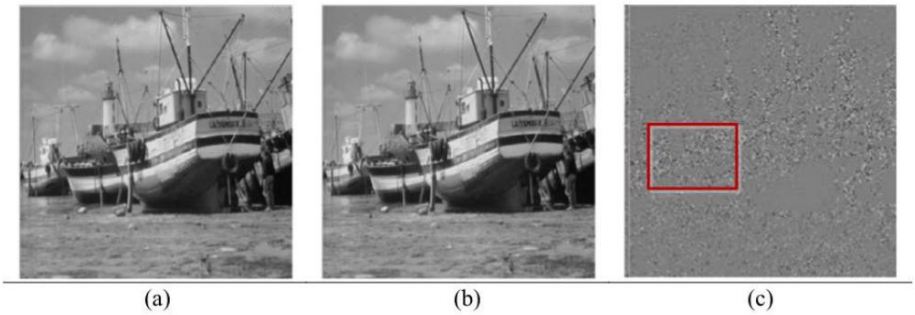


Figura 2.11. Ejemplo de una imagen portadora (a), su correspondiente inyectada por algoritmo de esteganografía HUGO con una carga de 0.4 bpp (b) y la sub-imagen seleccionada (c).

Rich models for steganalysis of digital images [Fridrich, 2012]: En este trabajo se presentó una técnica que es una de las más comunes para el estegoanálisis. Es una metodología general que fue basada en el concepto de modelo enriquecido espacial, que consistió en un gran número de submodelos. Los submodelos consideran varios tipos de relaciones entre muestras vecinas de residuos de ruido obtenidos por filtros lineales y no lineales.

El modelo enriquecido se ensambla como parte del proceso de aprendizaje y se basa en los ejemplos disponibles de imágenes limpias e inyectadas. Utilizaron un clasificador ensamblado para crear el modelo, por su baja complejidad computacional y su capacidad para trabajar de manera eficiente con espacios de características de alta dimensión y grandes conjuntos de entrenamiento.

Dado que la modificación de esteganografía en una imagen portadora sólo hace pequeños cambios en los pixeles, en este enfoque se modela sólo el componente de ruido (ruido residual) de las imágenes en lugar de su contenido.

Creación de submodelos

El objetivo general es capturar un gran número de diferentes tipos de dependencias entre los pixeles vecinos para dar al modelo la capacidad de detectar un amplio espectro de algoritmos de incrustación.

1. Cálculo de Residuos

Los submodelos se forman a partir de residuos de ruido (2.11).

$$\mathbf{R} = \left(R_{ij}\right) \in \mathbb{R}^{n_1 \times n_2} \quad (2.11)$$

Donde:

R es el residuo de ruido.

R_{ij} es el ruido en *i, j*.

$n_1 \times n_2$ es la dimensión de la imagen.

El residuo se calcula utilizando filtros paso alto con la ecuación (2.12).

$$R_{ij} = \hat{X}_{ij}\left(N_{ij}\right) - cX_{ij} \quad (2.12)$$

Donde:

c es el orden residual.

N_{ij} es un vecindario local del pixel X_{ij}, y $X_{ij} \notin N_{ij}$.

$\hat{X}_{ij}$ es el predictor de cX_{ij} definido en N_{ij}.

En la ecuación (2.12) se indica cómo se calculan los residuos de ruidos de la imagen procesada, esto se obtiene mediante la convolución. El predictor es un kernel.

De acuerdo con los autores, la gran ventaja de modelar el valor de ruido residual en lugar de los valores de pixeles, es que el contenido de la imagen se suprime en gran medida en ***R***, que tiene un rango dinámico mucho más estrecho, lo que permite una descripción estadística más compacta y robusta.

2. Truncamiento y Cuantificación.

Cada submodelo se forma a partir de una versión cuantificada y truncada del residuo:

$$R_{ij} \leftarrow trunc_T\left(round\left(\frac{R_{ij}}{q}\right)\right), \qquad (2.13)$$

donde: *round()* es la operación de redondeo a un entero. *q* es un paso de cuantificación.

En la ecuación (2.13), el propósito del truncamiento es reducir el rango dinámico del residuo de ruido para permitir su descripción mediante el uso de matrices de co-ocurrencia. La cuantificación hace que el residuo de ruido sea más sensible a los cambios de incrustación en las discontinuidades espaciales de la imagen (en bordes y texturas).

3. Generación de co-ocurrencias

La construcción de cada submodelo continúa con el cálculo de una o más matrices de co-ocurrencia de muestras vecinas a partir del residuo truncado y cuantificado.

Los autores analizaron 1000 imágenes portadoras del banco de imágenes BOSSbase v0.92, que las escogieron de forma aleatoria. Calcularon la correlación promedio entre los pixeles vecinos en las direcciones horizontal / vertical y diagonal / diagonal menor (ver Figura 2.12). Las correlaciones disminuyen gradualmente a medida que aumenta la distancia entre los pixeles y lo hacen más rápido para los pixeles adyacentes en diagonal. Por lo tanto, formaron co-ocurrencias de pixeles sólo a lo largo de las direcciones horizontales y verticales y evitando el uso de grupos con pixeles diagonales vecinos.

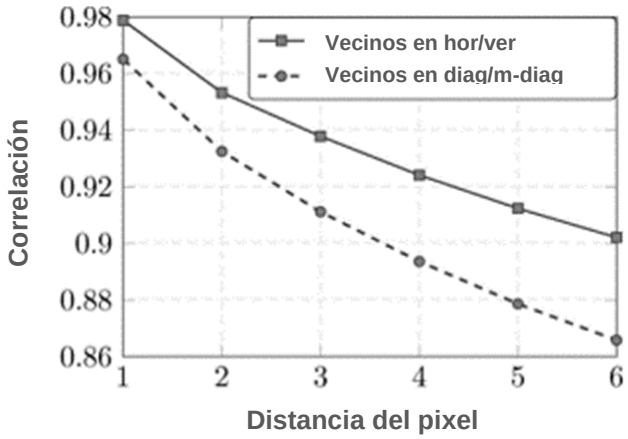


Figura 2.12. Correlación entre pixeles basada en su distancia [Fridrich, 2012].

En resumen, los submodelos se construyen a partir de co-ocurrencias horizontales y verticales de cuatro muestras residuales consecutivas procesadas usando (2.13) con *T* = 2 (*T* es un umbral). Formalmente, cada matriz de co-ocurrencia ***C*** es un vector de cuatro dimensiones indexado con ***d = (d1, d2, d3, d4)***, que da el vector con $(2\mathrm{T}+1)^4 = 625$ elementos.

El elemento de la coexistencia horizontal para el residuo $\boldsymbol{R = (R_{ij})}$ se define formalmente como el número (normalizado) de grupos de cuatro muestras residuales vecinas con valores iguales a *d1*, *d2*, *d3*, *d4*. La matriz de co-ocurrencia se obtiene con:

$$\boldsymbol{C}_{\boldsymbol{d}}^{(h)} = \frac{1}{Z}\left|\left\{\left(R_{ij}, R_{ij+1}, R_{ij+2}, R_{ij+3}\right),\ R_{ij+k-1} = d_k, k = 1, \ldots, 4\right\}\right| \qquad (2.14)$$

donde:

Z es el factor de normalización.

h indica la co-ocurrencia horizontal.

d_k es el índice de la dimensión y *k* es la dimensión.

La co-ocurrencia vertical (*v*) se define de manera similar. Finalmente, las características se unen de acuerdo con propiedades de simetría. El banco de

imágenes que se utilizó fue BOSSbase versión 0.92 [Filler & Pevny, 2010] que contiene 9074 imágenes en escala de grises.

Entrenaron el modelo con un clasificador ensamblado [Kodovsky, 2011] y con vectores de dimensión 12,753. Para el entrenamiento utilizaron 8074 imágenes y para prueba 1000. Probaron el modelo para detectar principalmente el algoritmo de inyección HUGO y el algoritmo Adaptable a los Bordes (EA*, Edge-Adaptive*) [Luo, 2010], como métrica usaron el error de detección P_E, los resultados se muestran en la Tabla 2.2.

Tabla 2.2. Rendimiento del SRM.

	HUGO	EA
Payload (bpp)	Media de P_E	Media de P_E
0.05	0.4240	0.3255
0.1	0.3940	0.2335
0.2	0.2658	0.1445
0.3	0.1915	0.0958
0.4	0.1355	0.0695

Al analizar la Tabla 2.2 se observa que el algoritmo de incrustación EA es más fácil de detectar al compararlo con el algoritmo HUGO. También, se muestra que la robustez de la característica del SRM, con su vector de dimensión de 12, 753 elementos, tuvo un comportamiento favorable al tener un error bajo en los *payloads* 0.2, 0.3 y 0.4 con el algoritmo HUGO**,** y cuando se detectó a HUGO con 0.05 y 0.1 bpp el error fue muy grande, indicando que no hubo distinción entre las clases, por el contrario para EA con todos *payloads* el error no es muy elevado.

Adaptive steganalysis against WOW embedding algorithm [Tang, 2014]: En este artículo se introdujo un enfoque en el cual incorporan la información del algoritmo de esteganografía para mejorar la capacidad de detección, es decir, se conoce la función de costo del algoritmo de inyección y su nivel de caga útil (*payload*) en bpp (bits por pixel).

En su enfoque se propuso una técnica de estegoanálisis como contramedida el algoritmo de esteganografía WOW (*Wavelet Obtained Weights*) mediante la

extracción de características de las regiones con altas probabilidades de modificación, que se encuentran ubicadas por la función de costo de WOW.

La idea principal fue que sabiendo los costos del algoritmo, se estimaron aquellas regiones donde el mensaje podría estar inyectado. Las regiones sospechosas se calculan con un parámetro *p* que indica la cantidad de pixeles seleccionados para extraer características y descartando aquellas zonas en donde no se modifican los pixeles. La figura 2.13 se ilustra el resultado del método.

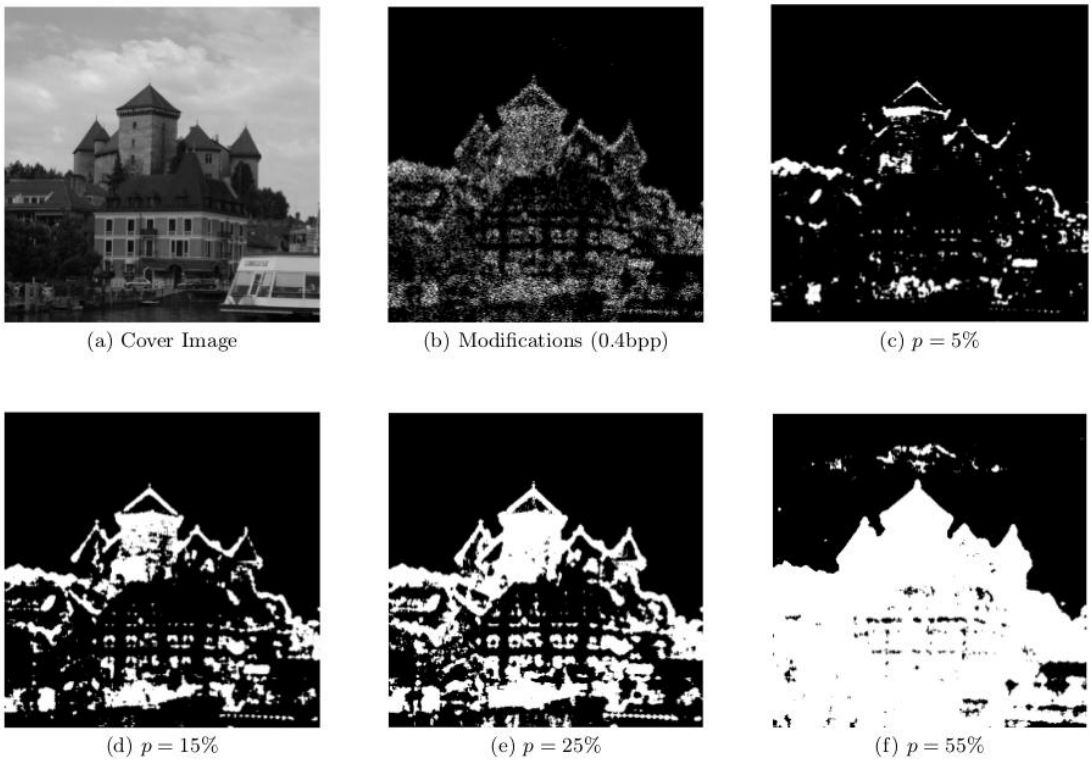


Figura 2.13. Modificaciones de pixeles (b), y sus respectivas regiones con diferentes valores de *p* [Tang, 2014].

Selection-channel-aware rich model for steganalysis of digital images [Denemark, 2014]: En este artículo se presentó una técnica inspirada por el artículo anterior [Tang, 2014]. Se propusieron dos características de estegoanálisis que llamaron maxSRM y maxSRMd2. La esteganografía adaptativa utiliza funciones de costo que están determinados por el contenido local de la imagen, lo que significa que el analista de estegoanálisis puede estimarlos a partir de la imagen inyectada.

Si el analista conoce el tamaño de la carga útil o si puede estimarlo, también puede estimar las probabilidades reales de cambio de inyección utilizadas por el emisor y organizar un ataque informado. La característica maxSRM se genera de la misma manera que el SRM, pero el proceso de formación de las matrices de co-ocurrencia se modifica para considerar las probabilidades de cambio de incrustación $\hat{\beta}$ estimadas a partir de la imagen analizada. En maxSRM se modifica la ecuación (2.14) para quedar como:

$$\widetilde{\boldsymbol{C}}_{\boldsymbol{d_0 d_1 d_2 d_3}} = \sum_{i,j=1}^{n_1,n_2-3} \max_{k=0,\dots,3} \hat{\beta}_{i,j+k} \left[r_{ij} = d_k, \forall k = 0, \dots, 3 \right]. \qquad (2.15)$$

Donde $\hat{\beta}_{i,j}$ es la probabilidad de incrustación en la posición *i,j+k.*

En este método, en lugar de agregar 1 a la bandeja correspondiente de la co-ocurrencia se agrega el máximo de las probabilidades de cambio de inyección tomadas entre los cuatro residuos de ruido. De esta forma, los grupos de cuatro pixeles con poca probabilidad de ser cambiados no afectarán mucho los valores de co-ocurrencia, mientras que aquellos en los que es probable que al menos un pixel cambie sí afectará. Los autores indican que el resto del proceso de formación del SRM se mantiene exactamente igual.

También se propuso otro componente de diseño del SRM, que es la dirección de exploración de co-ocurrencia. El SRM original usa escaneos horizontales y verticales (ver el caso "hv" en la Figura 2.14). Se estudiaron otras tres posibilidades que se muestran en la misma figura: direcciones diagonales y diagonales menores ("dm"), y dos direcciones "oblicuas" marcadas con "d2" y "dd".

Con base en los experimentos realizados concluyeron que las direcciones diagonales son las peores, mientras que las direcciones oblicuas son muy similares y dan mejores resultados (d2 y dd). Por lo tanto, decidieron incluir la versión del maxSRM con todas las direcciones de exploración de ocurrencia simultánea reemplazadas con la dirección oblicua "d2". Llamaron a esta versión del modelo enriquecido el maxSRMd2.

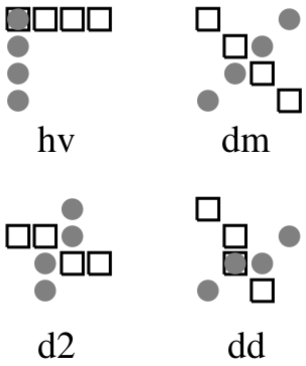


Figura 2.14. Direcciones de escaneo de cuatro tipos de co-ocurrencia.

El banco de imágenes que usaron fue BOSSbase versión 1.01 [Bas, 2011] que contiene 10,000 imágenes de tamaño 512x512 en escala de grises. Al crea el modelo usaron un clasificador ensamblado [Kodovsky, 2011]. Para el entrenamiento utilizaron 5000 imágenes y para prueba 5000. Probaron el modelo para detectar principalmente el algoritmo de inyección WOW y S-UNIWARD, descritos en la Sección 2.2.1.2.2, como métrica usaron el error de detección P_E, los resultados se muestran en la Tabla 2.3.

Tabla 2.3. Rendimiento con SRM y maxSRM.

Algoritmo	Técnica	0.05	0.1	0.2	0.3	0.4	0.5
WOW	SRM	.4572	.4026	.3210	.2553	.2060	.1683
	maxSRM	.3595	.3025	.2383	.1943	.1623	.1371
	maxSRMd2	.3539	.2997	.2339	.1886	.1543	.1306
S-UNIWARD	SRM	.4533	.4024	.3199	.2571	.2037	.1640
	maxSRM	.4209	.3684	.2981	.2431	.1992	.1633
	maxSRMd2	.4180	.3660	.2886	.2360	.1908	.1551

En la Tabla 2.3 se observa que las características maxSRM y maxSRMd2 al utilizar mayor información del algoritmo de inyección, tuvieron un mejor desempeño con ambos algoritmos que el SRM. Con el algoritmo WOW pudieron detectar el *payload* 0.2, mientras que el SRM no pudo distinguir las clases. Con S-UNIWARD ocurrió el mismo comportamiento, pero se muestra que éste algoritmo resiste mejor al estegoanálisis utilizando las características maxSRM y maxSRMd2.

Adaptive steganalysis based on embedding probabilities of pixels [Tang, 2016]: En este artículo cuestionan que algunos métodos de estegoanálisis, como el Modelo Enriquecido Espacial (SRM) no consideran las regiones más probables a una inyección y que procesan toda la imagen por igual. Asumiendo que todos los pixeles contribuyen igualitariamente al extraer las características de estegoanálisis.

Por tal motivo en su trabajo fue propuesta una técnica que consideró una mayor atención a aquello pixeles con altas probabilidades de cabio de inyección. Calcularon las regiones de probabilidades de inyección y asignaron pesos a cada pixel, generando el promedio para una vecindad de pixeles dada, con la ecuación:

$$w_{i,j}^{(h)} = \frac{p_{i,j} + p_{i,j+1} + p_{i,j+2} + p_{i,j+3}}{4}, \qquad (2.16)$$

donde $p_{i,j}$ es el valor del pixel, *h* = indica que es horizontal.

Después compararon el valor de los residuos y el peso de los pixeles para generar una matriz de co-ocurrencia con la ecuación:

$$\bar{C}_d^{(h)} = \frac{1}{z}\sum_{i=1}^{N}\sum_{j=1}^{M-3} w_{i,j}^{(h)} \times \left|(r_{i,j}, r_{i,j+1}, r_{i,j+2}, r_{i,j+3}) = (d_1, d_2, d_3, d_4)\right| \qquad (2.17)$$

donde *Z* es el factor de normalización, *N* y *M* son las dimensiones de la imagen. Sin embargo los autores no fueron muy claros al utilizar la métrica de evaluación de su técnica.

Improving selection-channel-aware steganalysis features [Denemark, 2016]: En este artículo se realizó una mejora de [Denemark ,2014]. Fue propuesta una característica llamada σmaxSRM una versión mejorada de maxSRM, reemplazando las probabilidades de cambio de inyección estimadas de los pixeles con la distorsión de los residuos obtenidos de la imagen.

Se expuso que existe una discrepancia en maxSRM en el sentido de que se acumulaban solamente las probabilidades de cambio de incrustación de pixeles en

los contenedores de co-ocurrencia de residuos. Por lo tanto, modificaron las tasas de cambio del pixel por una medida de la distorsión residual.

El residuo lineal $\boldsymbol{Z}$ en SRM se obtiene mediante la convolución de la imagen con un kernel, como éste descriptor es una variante del SRM [Fridrich, 2012], implementa los mismos, un ejemplo es la ecuación (2.18).

$$\boldsymbol{K} = [-1 \;\; 1] \tag{2.18}$$

$$\boldsymbol{Z}^{(SRM)}(\boldsymbol{X}) = \boldsymbol{K} \star \boldsymbol{X} \tag{2.19}$$

Y en coordenadas:

$$\mathcal{Z}_{i,j}^{(SRM)}(\boldsymbol{X}) = \sum_{k,l} K_{kl} x_{i-k,j-l.} \tag{2.20}$$

Donde: *ij* son índices del residuo.

kl son índices de un kernel.

En resumen esta técnica, se adapta de maxSRM reemplazando el valor máximo de las probabilidades de cambio de incrustación estimadas, con la diferencia esperada en el residuo filtrado para calcular las co-ocurrencias. La matriz de co-ocurrencia de cuarto orden en su forma horizontal se calcula de la siguiente forma.

$$C_{d_0,d_1,d_2,d_3}^{\sigma maxSRM} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2-3} \max\left(\sigma_{i,j}, \sigma_{i,j+1}, \sigma_{i,j+2}, \sigma_{i,j+3}\right) \times [(r_{i,j}, r_{i,j+1}, r_{i,j+2}, r_{i,j+3}) = (d_1, d_2, d_3, d_4)] \tag{2.21}$$

Donde $\sigma_{i,j}$ es un valor estimado de la diferencia esperada en el residuo, se obtiene con:

$$\sigma_{i,j} = E\left(\left|\mathcal{Z}_{i,j}(\boldsymbol{Y}) - \mathcal{Z}_{i,j}(\boldsymbol{X})\right|\right). \tag{2.22}$$

Que puede ser estimado para un filtro lineal con la ecuación (2.23).

$$\sigma_{i,j} = \sqrt{\sum_{k,l} w_{k,l}^2 \beta_{i-k,j-l}}, \quad (2.23)$$

donde $w_{k,l}$ es el peso (coeficiente) del filtro lineal, β_{ij} es la probabilidad de cambio de incrustación.

Sin embargo, para los residuos de ruido no lineales, se puede estimar utilizando la simulación de Monte-Carlo con incrustaciones repetidas, lo que es computacionalmente costoso, y los autores no explican lo suficiente.

El banco de imágenes que usaron fue BOSSbase versión 1.01 [Bas, 2011]. La dimensión del vector de maxSRMq2d2 (d2 se refiere al escaneo) es de 12,753 elementos. Utilizaron un clasificador ensamblado [Kodovsky, 2011]. Para el entrenamiento utilizaron 5000 imágenes y para prueba 5000. Probaron el modelo para detectar principalmente el algoritmo de inyección HILL y WOW, descritos en la Sección 2.2.1.2.2, como métrica usaron el error de detección P_E, los resultados se muestran en la Tabla 2.4.

Tabla 2.4. Rendimiento de la técnica sigma maxSRM y maxSRM.

Algoritmo	0.2 bpp Media de $\mathbf{P_E}$	0.40 bpp Media de $\mathbf{P_E}$
HILL		
maxSRMq2d2	0.3181	0.2238
σmaxSRMq2d2	0.3075	0.2132
WOW		
maxSRMq2d2	0.2472	0.1658
σmaxSRMq2d2	0.2449	0.1620

Al analizar la Tabla 2.44 se observa que la característica que propusieron σmaxSRMq2d2 aumenta el nivel de clasificación y por lo tanto, el error de detección decremento, pero no de forma notable. Con el algoritmo HILL mejoro un 1.06% con ambos *payloads*. Con el algoritmo WOW mejoro un 0.23% y 0.38 con 0.2 bpp y 0.40 bpp, respectivamente.

Adaptive Steganalysis via Augmented Utilization of Selection-Channel Information [Zhou, 2018]: Es este trabajo se diseñó una técnica para analizar de forma aumentada los pixeles con altas probabilidades de cambio de inyección, para contribuir más en la extracción de características. Utilizaron una función para fortalecer las probabilidades de cambio de incrustación, llamando a estos coeficientes aumentados. Los coeficientes aumentados son ponderados por las probabilidades aproximadas de los residuos de la imagen, y se utilizaron para calcular las características de la matriz co-ocurrencia de cuarto orden. Llamaron a su enfoque αSRM.

Expusieron que, tomando como referencia la técnica maxSRM, los autores consideraron hacer un mejor uso de la información del canal informado al fortalecer las probabilidades de cambio de incrustación estimadas para obtener coeficientes aumentados y formar características de co-ocurrencia con coeficientes aumentados y los residuos. En la figura 2.15 se muestra el diagrama de bloques.

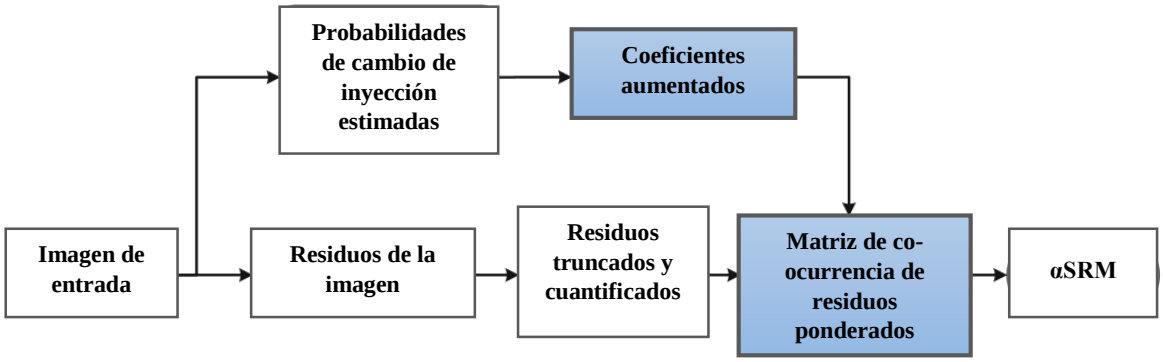


Figura 2.15. Diagrama de bloques de la técnica para extraer características [Zhou, 2018].

a) Fortalecimiento de las probabilidades de cambio de incrustación

Se fundamentaron en que los pixeles con altas probabilidades de modificaciones de inyección pueden contribuir más a las características de estegoanálisis. Para este motivo utilizaron un filtro paso alto ***H***. Se calcula con la ecuación (2.24).

$$\boldsymbol{\mathcal{E}} \triangleq \left(\mathcal{E}_{i,j}\right)^{n_1 \times n_2} = \boldsymbol{\beta} \otimes \boldsymbol{H} \quad (2.24)$$

donde:

$\boldsymbol{\beta} = \left(\beta_{i,j}\right)^{n_1 \times n_2}$ es una matriz de probabilidades de inyección.

$\otimes$ es la operación de convolución.

Aumentaron las probabilidades de cambio de inyección (coeficiente aumentado) mediante:

$$\alpha_{i,j} = \beta_{i,j} + \lambda\left|\mathcal{E}_{i,j}\right|, \qquad 1 \le i \le n_1, 1 \le j \le n_2 \quad (2.25)$$

donde: λ> 0 es el parámetro para controlar la fuerza de aumento y el símbolo | · | es el operador de tomar el valor absoluto.

b) Formación las características de co-ocurrencia de residuos ponderados

En lugar de utilizar los coeficientes aumentados directamente, los autores usaron los coeficientes ponderados por las probabilidades de ocurrencia de residuos de imagen. Se suele suponer que los residuos de la imagen siguen una distribución Laplaciana, en la que los valores pequeños tienen grandes probabilidades. Para simplificar el cálculo, realizaron algunos experimentos y aproximaron la distribución con un modelo no paramétrico fijo de la siguiente manera:

$$Pr_{i,j} = \begin{cases} 0.4, & r_{i,j} = 0, \\ 0.2, & r_{i,j} = \pm 1, \\ 0.1, & r_{i,j} = \pm 2, \end{cases} \quad (2.26)$$

donde: $r_{i,j}$ es el residuo de ruido.

La característica de co-ocurrencia horizontal de este enfoque αSRM se calcula como:

$$C^{\alpha SRM}_{d_0,d_1,d_2,d_3} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2-3}\left(\sum_{k=0}^{3}\alpha_{i,j+k}\,Pr_{i,j}\right)\times[(r_{i,j},r_{i,j+1},r_{i,j+2},r_{i,j+3}) = (d_1,d_2,d_3,d_4)] \quad (2.27)$$

donde:

$\alpha_{i,j}$ es el coefiiente aumentado.

$Pr_{i,j}$es la probabilidad el residuo de ruido.

El banco de imágenes que manipularon fue BOSSbase versión 1.01 [Bas, 2011]. Probaron el modelo para detectar principalmente el algoritmo de inyección S-UNIWARD, HILL y MiPOD, descritos en la Sección 2.2.1.2.2. Utilizaron un clasificador ensamblado [Kodovsky, 2011]. Para el entrenamiento utilizaron 5000 imágenes y para prueba 5000, como métrica usaron el error de detección P_E, los resultados se muestran en la Tabla 2.5.

Tabla 2.5. Rendimiento de αSRM, maxSRM y SRM, con payloads de 0.1 a 0.5.

Algoritmo	Característica	0.1 bpp	0.2 bpp	0.3 bpp	0.4 bpp	0.5 bpp
S-UNIWARD	SRM	0.4022	0.3208	0.2561	0.2068	0.1636
	maxSRM	0.3697	0.2990	0.2440	0.1989	0.1638
	αSRM	0.3625	0.2869	0.2316	0.1866	0.1499
HILL	SRM	0.4324	0.3608	0.3002	0.2492	0.2051
	maxSRM	0.3803	0.3167	0.2688	0.2253	0.1897
	αSRM	0.3533	0.2875	0.2380	0.1973	0.1664
MiPOD	SRM	0.4165	0.3459	0.2885	0.2388	0.1998
	maxSRM	0.4011	0.3313	0.2792	0.2330	0.1942
	αSRM	0.4006	0.3331	0.2767	0.2276	0.1889

En la Tabla 2.5 se presenta la característica propuesta αSRM y se compara con las características: SRM y maxSRM, todas ellas con vectores de 34,671 elementos. El rendimiento es mejor en: maxSRM y αSRM, con respecto a SRM, esto se logró ya que ambas características utilizan más información del algoritmo a detectar. En la mayoría de los casos, αSRM funciona mejor.

2.3 Esteganografía y criptografía

La esteganografía pude confundirse con la criptografía, sin embargo existen unas diferencias notables. Al establecer una comunicación entre un emisor y un receptor, podrían generarse inconvenientes, debido a que existen factores que pueden irrumpir o atacar de forma intencional o no durante la comunicación en un canal público. Por ejemplo, al comunicarse por medio de internet.

Debido a esto, surgieron mecanismos que permitieran una comunicación de forma segura para proteger la información que se envía. Para esto, se recurrió a la criptografía y la esteganografía. Al utilizar la esteganografía, el mensaje se envía en un objeto portador, de tal manera que pase desapercibido y que solo sea del conocimiento por las partes que se comunican, es decir, la comunicación está oculta. Para el caso de la criptografía, al aplicarla convierte el mensaje en un formato ilegible, lo que permite que se identifique más fácilmente su uso en comparación la esteganografía.

La criptografía se define como el arte y la ciencia de transformar datos en una secuencia de bits que parece aleatoria y sin sentido para un observador secundario [Joseph, 2011]. La contramedida de la criptografía es el criptoanálisis que es la ingeniería inversa de la criptografía: intenta identificar las debilidades de varios algoritmos criptográficos y sus implementaciones para explotarlos. Cualquier intento de criptoanálisis se define como un ataque [Manoj, 2010]. Las diferencias entre criptografía y esteganografía se muestran en la Tabla 2.6.

Tabla 2.6. Diferencias entre criptografía y esteganografía [Mishra, 2015].

Criptografía	Esteganografía
Es una técnica para convertir el mensaje secreto en otra forma que no sea legible por humanos.	Es una técnica para ocultar la existencia de la comunicación.
Es un tipo de comunicación conocida.	Es un tipo de comunicación oculta.
La criptografía la altera estructura general de los datos.	La esteganografía no altera la estructura general de los datos.
El resultado final obtenido se conoce como texto cifrado.	El resultado final obtenido es conocido como un estego media.
Una vez que se ha descubierto, nadie puede obtener fácilmente los datos secretos.	Una vez que se ha descubierto, solo aquellos que conozcan el mecanismo de inyección y la llave pueden obtener los datos secretos.

2.4 Discusión

Como se presentó en la sección anterior, en el estado del arte abordan el problema de la esteganografía de dos formas: las técnicas clásicas y las técnicas con características de alta dimensión.

a) Estegoanálisis en su forma clásica

Las técnicas clásicas plantean soluciones al problema por medio de estadística, atacan métodos como el LSB, y mecanismos derivados de éste, como el LSBM. El problema surge cuando se quiere detectar esteganografía adaptativa, que minimiza la distorsión de la imagen al inyectar el mensaje.

Dando como resultado que dichas técnicas no pueden diferenciar entre una imagen limpia y una inyectada. Como consecuencia solo se pueden atacar a la esteganografía clásica. Sin embargo, siguen publicándose artículos de estegoanálisis en su forma clásica, dejando el problema de la esteganografía adaptativa sin resolver.

b) Estegoanálisis con características de alta dimensión

La otra parte del estegoanálisis son aquellas técnicas que surgen como contramedida a la esteganografía adaptativa, las cuales se crean con un gran número de características al generar sus vectores. Este tipo de enfoques detectan de forma precisa los métodos de esteganografía clásica.

Para este propósito se necesitan formar características de gran dimensión, típicamente un descriptor global que considere los residuos filtrados de ruido y los agregue mediante co-ocurrencias de la imagen procesada. En [Fridrich, 2011] el descriptor tiene dimensión de 33,963. En los trabajos de [Fridrich, 2012], [Denemark, 2014] y [Zhou, 2018] la dimensión es de 34,671 y una versión reducida de 12,753. En [Chen, 2013] es de 6,000. Lo que provoca que se cree un costo computacional elevado al calcular las características (se muestra el tiempo del cálculo en la sección 5.4). Una posible oportunidad de mejora es establecer un balance entre el número de características (dimensión) y el nivel de la detección. Es

importante mencionar que los trabajos revisados no existe un método capaz de detectar con una eficacia del 100%.

c) Banco de imágenes

Se observó que el banco de imágenes BOSSbase versión 10.1 [Bas, 2011], se utiliza en la mayoría de los trabajos revisados. Cuenta con 10,000 imágenes limpias en formato bruto PGM, de tamaño 512x512 y en escala de grises.

d) Esteganografía

Para los investigadores que trabajan con la esteganografía adaptativa, no solo se deberían limitar a diseñar una función de costo. Para mejorarla, se pueden crear nuevos artificios que manipulen la imagen, tal es el caso de [Li, 2015] en donde utiliza una función de costo ya creada, que se suma a su enfoque y como resultado aumenta la seguridad.

2.5 Propuesta de solución

En esta sección se describe de forma rápida la propuesta de solución que será retomada en el próximo capítulo. Se describe el objetivo, los alcances y limitaciones de este trabajo de investigación.

2.5.1 Descripción de la propuesta de solución

Como se mencionó en el apartado (a) de la sección 2.4, si bien en la esteganografía clásica se siguen proponiendo mejoras, en este trabajo de tesis se decidió trabajar con el problema de la esteganografía adaptativa, porque tiene mayor relevancia y es un problema que aún no se ha resuelto por completo (detección al 100%). Además, el número de técnicas de esteganografía reportadas en el estado del arte son mayores en comparación con las técnicas de estegoanálisis.

Se revisó que el descriptor más utilizado y frecuente, debido a su capacidad para detectar a métodos de esteganografía adaptables, es el Modelo Enriquecido Espacial (SRM, *Spatial Rich Model*). Esta técnica ha sido modificada por diversos autores, en algunos casos presenta mejoras y en otros casos tiene resultados variables. Sus variantes demuestran una tendencia que permite oportunidades para aprovechar este método.

Para el caso del clasificador, se revisó que el clasificador SVM (*Support Vector Machine*) con un kernel Gaussiano y el Clasificador Ensamblado que implementa como clasificador el Discriminante Lineal de Fisher, son los más utilizados, pero el rendimiento de este último es mejor.

Para las técnicas de esteganografía las más utilizadas son: HUGO, WOW, S-UNIWARD y HILL. En práctica, por la gran cantidad de imágenes procesadas los algoritmos se implementan inyectando un mensaje aleatorio en las imágenes.

En conclusión, tomando como referencia los aspectos descritos anteriormente, y con el fin de cumplir con el objetivo planteado en esta tesis se seleccionaron las siguientes técnicas:

- Estegoanálisis
    - SRM
    - Clasificador ensamblado
- Esteganografía
    - WOW
    - HILL

Con las técnicas seleccionadas se propuso un método para obtener una sub-imagen, que sumado con éstas pueda detectar si una imagen está inyectada o es una imagen limpia. En el siguiente capítulo se explica a detalle el método propuesto.

2.5.2 Objetivo

Estudiar técnicas de IA aplicables al estegoanálisis de imágenes digitales, y proponer un método o algoritmo desde un enfoque de IA para detectar imágenes inyectadas.

2.5.3 Alcances y limitaciones

El presente trabajo de investigación se sujetó a las restricciones siguientes:

- Usar o adaptar técnicas de esteganografía para crear bancos de imágenes.
- Evaluar los algoritmos y técnicas de IA y sus enfoques para la detección de esteganografía.
- Escoger al menos tres técnicas de Inteligencia Artificial, aplicables al estegoanálisis de imágenes.
- Proponer e implementar un método para identificar imágenes inyectadas.
- El método sólo funcionará con imágenes digitales.
- Detectar imágenes inyectadas con dos de las técnicas más frecuentes de esteganografía.
- Las imágenes fueron inyectadas con texto.

- Dado que se trabajó con el banco de imágenes BOSSBase 1.01 (Sección 2.4 de la Propuesta de tesis), las características de las imágenes son las siguientes. Escala de grises y formato pgm.

2.6 Conclusiones

En este capítulo, para comprender el problema que se abordó se presentó el contexto teórico y los trabajos relacionados con la tesis. Además, se describieron las técnicas de esteganografía y su contramedida. Con la información presentada se concluye:

- Las técnicas de estegoanálisis son superadas por las técnicas de esteganografía.
- El *framework* STC (*Syndrome-Trellis Codes*) es el más usado en los últimos años ya que minimiza la distorsión y es reversible. Se comparte la Matriz ***H*** como llave entre el emisor y receptor.
- La esteganografía puede mejorar en la seguridad al aplicarse un algoritmo de criptografía antes de incrustar el mensaje, de esta forma incluso cuando se lograra extraer el mensaje éste estaría encriptado.
- Cuando se trabaja con esteganografía adaptable al contenido, esta no debe de limitarse a solo diseñar una función de costo sino buscar mecanismos para trasformar la imagen, un ejemplo el trabajo de [Li, 2015].
- Los algoritmos seleccionados para detectar en este trabajo fueron: WOW y HILL. Las técnicas seleccionadas para estegoanálisis fueron: SRM y el Clasificador Ensamblado que implementa como clasificador el Discriminante Lineal de Fisher.

Capítulo 3

Propuesta de solución

3.1 Análisis del problema

El problema descrito en esta tesis radica en detectar dos de las técnicas más frecuentes de esteganografía, específicamente etiquetar si una imagen está inyectada o no. Es decir, es un problema donde solo existen dos clases: limpia e inyectada, una clasificación binaria.

Para resolver este problema, el método propuesto hace uso de técnicas de IA para generar características, que sean capaces de separar las clases, y generar

un modelo que permita realizar la clasificación. Como se revisó en el estado del arte, en ningún caso expuesto en este trabajo se trata de recuperar el mensaje embebido y solo se realiza la clasificación.

Para la inyección de mensajes de texto, por su rápida implementación el algoritmo HILL fue implementado. Para la generación de bancos de imágenes con los dos algoritmos que fueron seleccionados, el mensaje que fue incrustado es un mensaje aleatorio.

3.1.1 El reto de la esteganografía adaptativa

La esteganografía es la habilidad de transmitir información a través de objetos portadores como imágenes. No obstante, debido a la mínima distorsión que se genera al utilizar la codificación STC más una función de costo. Los cambios se hacen el aquellos pixeles en donde hay zonas ideales para la inyección, es decir, que no producen cambios significativos en la imagen, como zonas de textura o con ruido.

El problema es que muchas veces no se puede distinguir entre una inyección y el ruido normal de una imagen. El caso más deseable es cuando en el resultado de clasificación es un Falso Positivo, es decir, que una imagen limpia se haya predicho como inyectada. El peor de los casos se genera cuando hay muchos Falsos Negativos, en consecuencia indicaría que una imagen que está inyectada se ha predicho como limpia.

3.1.2 Esteganografía adaptativa como un problema de clasificación

Como se describió en el aparado 2.2.2 las mejores técnicas que han sido reportadas son construidas mediante características de estegoanálisis y aprendizaje automático.

Generalmente se tiene un banco de imágenes limpias y el algoritmo o algoritmos a detectar son conocidos, permitiendo que se genere un banco de imágenes inyectadas. Por último se entrena a un clasificador binario para separar

las clases. Hay dos componentes en este enfoque las características y el algoritmo de clasificación [Ker, 2013]. En la figura 3.1 se muestra el diagrama del enfoque.

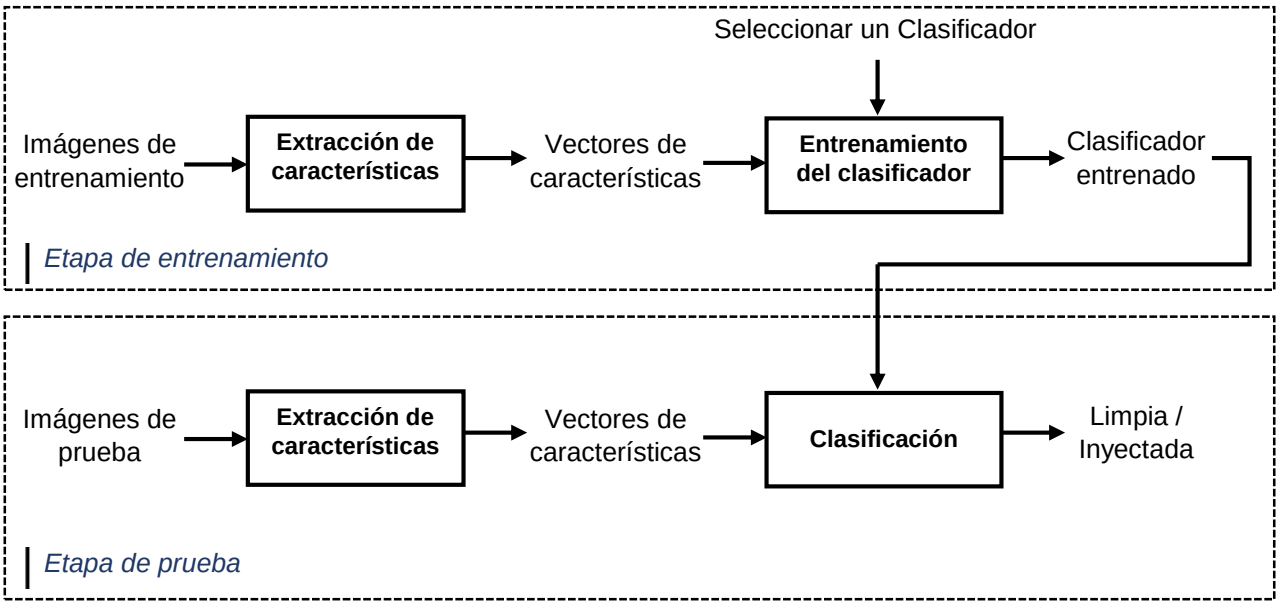


Figura 3.1. Diagrama de Clasificación Binaria en Estegoanálisis.

3.2 Propuesta de solución

En esta sección se describe la metodología para generar un modelo de aprendizaje de estegoanálisis para hacer una clasificación. El objetivo principal de la esteganografía adaptable al contenido es embeber un mensaje en zonas que son difíciles de modelar (texturas y/o bordes) en donde el cambio del último bit del pixel no genera una distorsión significativa en la imagen.

Consecuentemente se producirá una región de la imagen en la cual se concentre la mayor parte del mensaje. De acuerdo con lo reportado en la literatura, cuando se tiene una mayor carga de mensaje (bpp) el nivel de detección mejora. Con estas dos observaciones, el objetivo es obtener la mejor zona de la imagen, a la cual se le denomina Sub-imagen que podría generar resultados cercanos o mejores a los reportados en la literatura.

Primero se emplea el proceso para obtener la mejor sub-imagen de una imagen procesada, para después aplicar el SRM [Fridrich & Kodovsky, 2012] y utilizar el clasificador ensamblado [Kodovsky & otros, 2011]. En la Figura 3.2 se muestra el diagrama de la metodología propuesta.

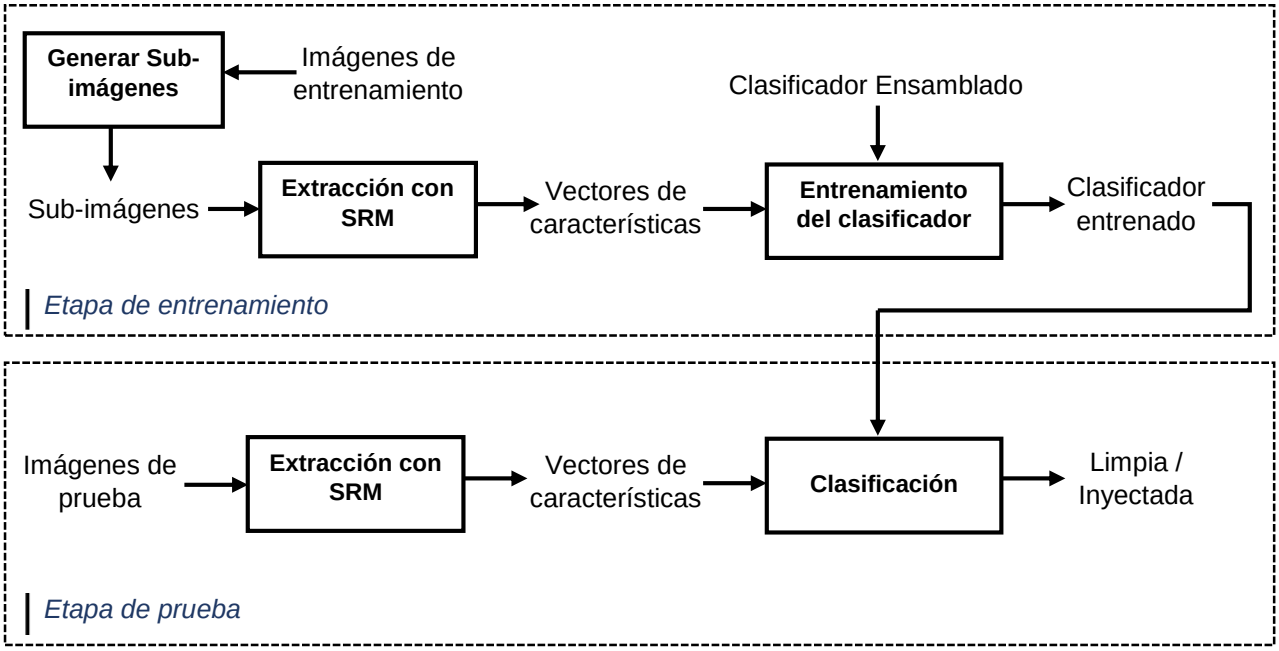


Figura 3.2. Metodología propuesta.

3.2.1 Detección de técnicas de esteganografía

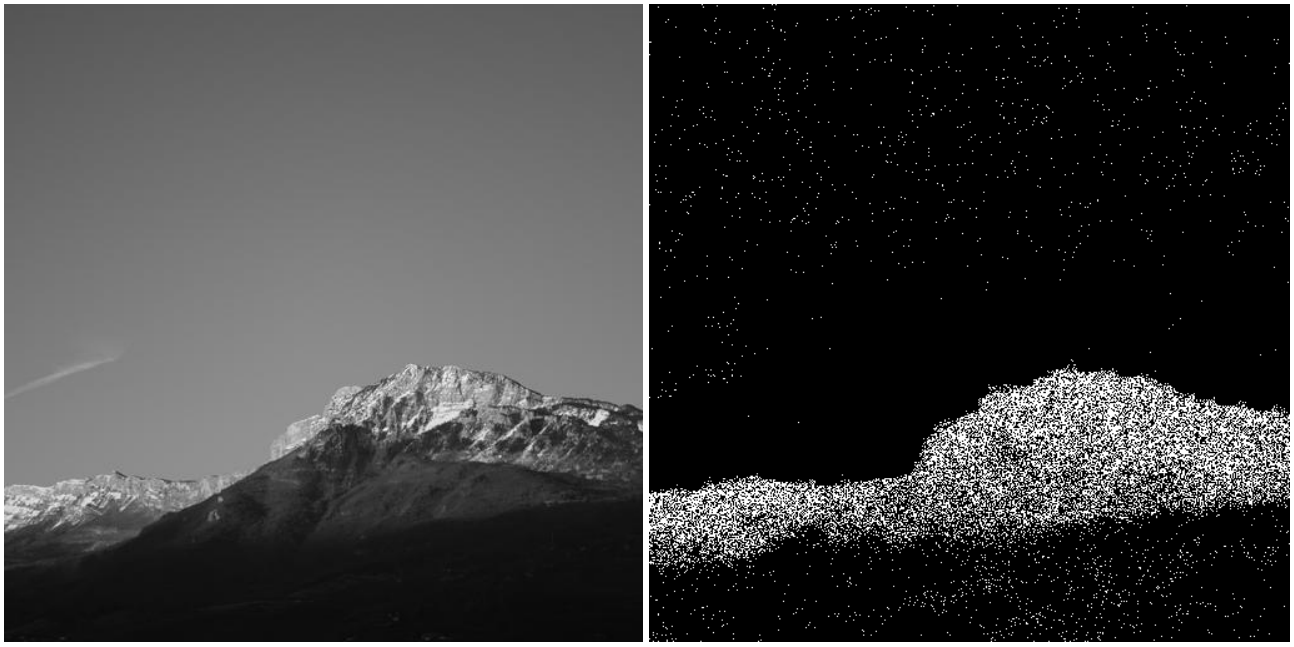
En este apartado se muestran las zonas modificadas al inyectar un mensaje aleatorio de los dos algoritmos seleccionados. La imagen seleccionada se utilizó con ambos algoritmos.

3.2.1.1 Función de costo HILL

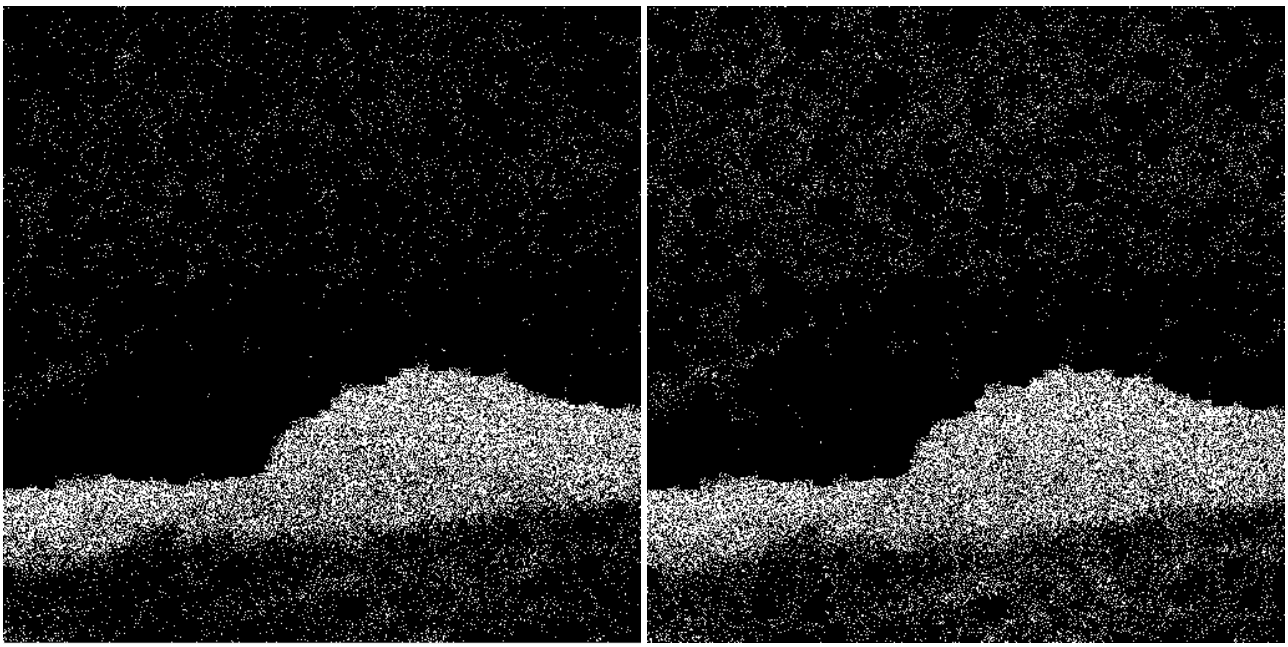
En la figura 3.3 se muestran los pixeles modificados con el algoritmo HILL con payloads de 0.3, 0.4 y 0.5 bpp.

3.2.1.2 Función de costo WOW

En la figura 3.4 se muestran los pixeles modificados con el algoritmo WOW con payloads de 0.3, 0.4 y 0.5 bpp.



(a) (b)



(c) (d)

Figura 3.3. Incrustación con el algoritmo HILL, (a) es la imagen portadora, (b) con 0.30 pbb, (c) con 0.40 bpp y (d) con 0.50 bpp.

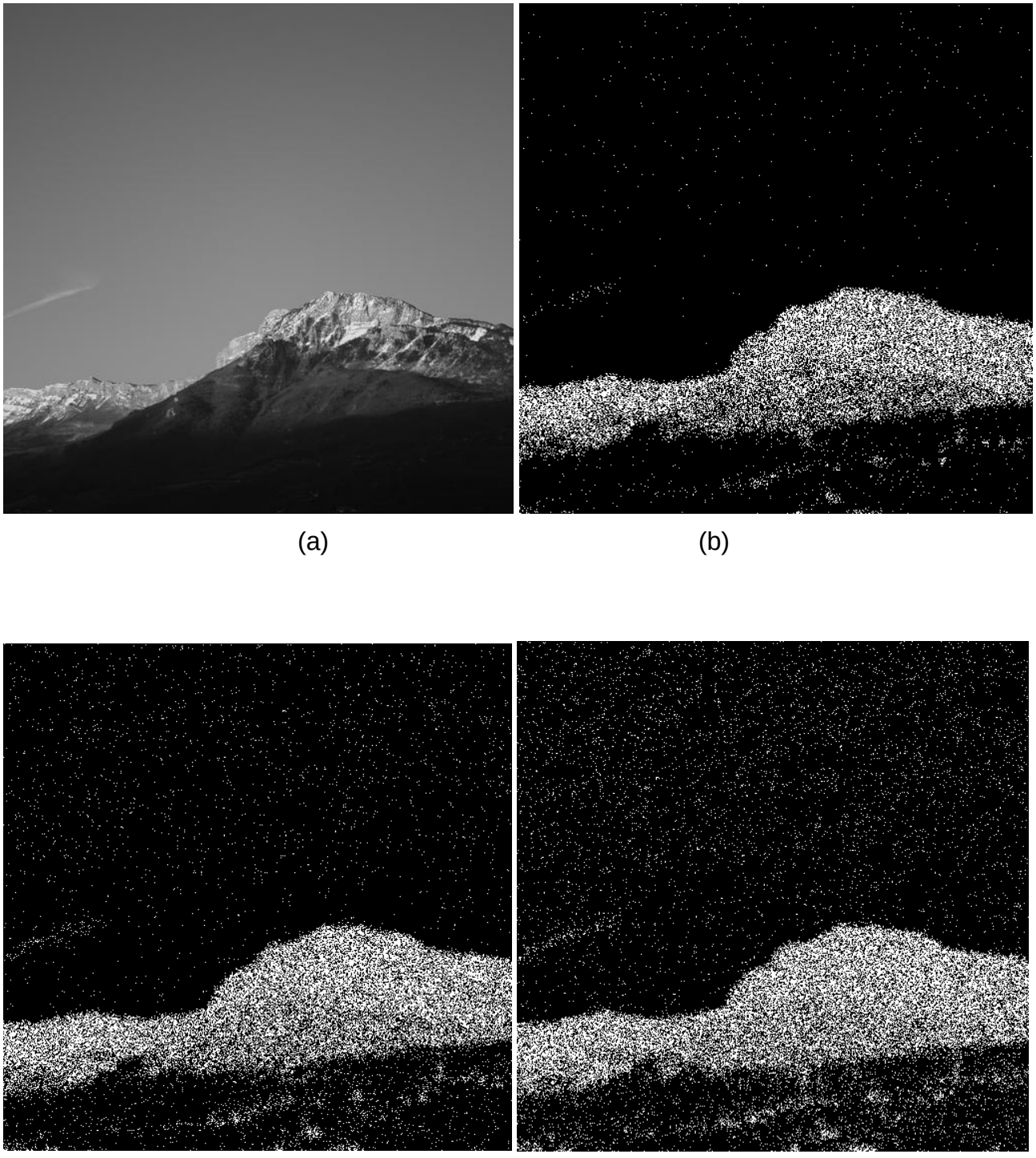


Figura 3.4. Incrustación con el algoritmo WOW, (a) es la imagen portadora, (b) con 0.30 pbb, (c) con 0.40 bpp y (d) con 0.50 bpp.

3.2.2 Técnicas de Inteligencia Artificial

3.2.2.1 Descriptores a utilizar

Tomando en cuenta los descriptores reportados en la literatura y atendiendo el objetivo, los alcances y limitaciones de esta tesis, se utilizó el descriptor global SRM, que cuenta con un vector de características de alta dimensión, es decir tiene 34,671 características para describir una imagen. Esto se debió a que los descriptores para un sistema de Visión Artificial en [Gonzalez, 1977], no son efectivos para la detección.

Sin embargo, se implementaron los descriptores: SPAM (Subtractive Pixel Adjacency Model)[Pevny, 2010] y los descriptores GLCM (Gray level Co-occurance Matrix) [Haralick, 1973], para futuras investigaciones.

3.2.2.2 Proceso para generar Sub-imágenes

A continuación explica cómo se genera la Sub-imagen, mostrado en diagrama de la Figura 3.2.

a) Mapa de inyección

El primer paso es calcular el mapa de inyección o mapa de cambio, éste se calcula al aplicar la diferencia entre dos imágenes en este caso la imagen limpia y la inyectada, se calcula con la ecuación:

$$MI = \sum_{i=1}^{n1} \sum_{j=1}^{n2} \left| Y_{ij} - X_{ij} \right|. \qquad (3.28)$$

Donde ***MI*** es el mapa de inyección, X_{ij} es la imagen limpia, Y_{ij} es su correspondiente inyectada, con una dimensión de *n1xn2.*

b) Reducción de dimensión

En el paso anterior se genera una imagen binaria, el siguiente paso es reducir el espacio de búsqueda de la región, para esto se reduce el mapa de inyección de tamaño *MxN* a un tamaño $(M \times tR)$ x $(N \times tR)$, donde *tR* es un umbral de reducción.

Se utiliza el método de interpolación bilineal [Gonzalez, 1977], [Bovik, 2009]. Dadas cuatros coordenadas vecinas de la imagen $f(n_{10}, n_{20}), f(n_{11}, n_{21}) f(n_{12}, n_{22})$ y $f(n_{11}, n_{23})$, la imagen trasformada se calcula como:

$$g(n_1, n_2) = A_0 + A_1 n_1 + A_2 n_2 + A_3 n_1 n_2 \quad (3.29)$$

que es una función bilineal en las coordenadas (n_1, n_2). Los pesos A_0, A_1, A_2 y A_3 se obtienen resolviendo:

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 1 & n_{10} & n_{20} & n_{10}n_{20} \\ 1 & n_{11} & n_{21} & n_{11}n_{21} \\ 1 & n_{12} & n_{22} & n_{12}n_{22} \\ 1 & n_{13} & n_{23} & n_{13}n_{23} \end{bmatrix}^{-1} \begin{bmatrix} f(n_{10}, n_{20}) \\ f(n_{11}, n_{21}) \\ f(n_{12}, n_{22}) \\ f(n_{13}, n_{23}) \end{bmatrix}. \quad (3.30)$$

Por lo tanto, $g(n_1, n_2)$ se define como la combinación lineal de los niveles de gris de sus cuatro vecinos más cercanos.

c) Búsqueda de la región

Una vez realizada la reducción se procede a encontrar la mejor partición, es decir, la porción que posea la mayor área de la imagen, el área se calcula con (3.31).

La sub-imagen con mayor área se calcula con la ecuación (3.30):

$$S(MIR) = max\left(\sum_{i=1}^{M_1}\sum_{j=1}^{N_2} A(MIR_{ij})\right) \quad (3.31)$$

$$A(r) = \sum_{i=1}^{a}\sum_{j=1}^{b} r_{ij}. \quad (3.32)$$

Donde M_1 y N_1 son las dimensiones del mapa de inyección reducido. La ecuación (3.31) calcula el área de una determinada región *r*. La dimensión de *a* y *b* son la mitad de M_1 y N_2, respectivamente. Para acelerar el proceso de búsqueda se

hace un *salto* en los índices *i* y *j* de la ecuación (3.30), es decir, cuando *i* y *j* se incrementan lo hacen sumado el valor de ***salto***.

Como último paso la función *S(MIR)* en (3.30), retorna las posiciones *ij* las cuales se escalan a la imagen original y se recorta la imagen limpia e inyectada desde las posiciones hasta una dimensión específica. La figura 3.5 y 3.6 ilustra el proceso.

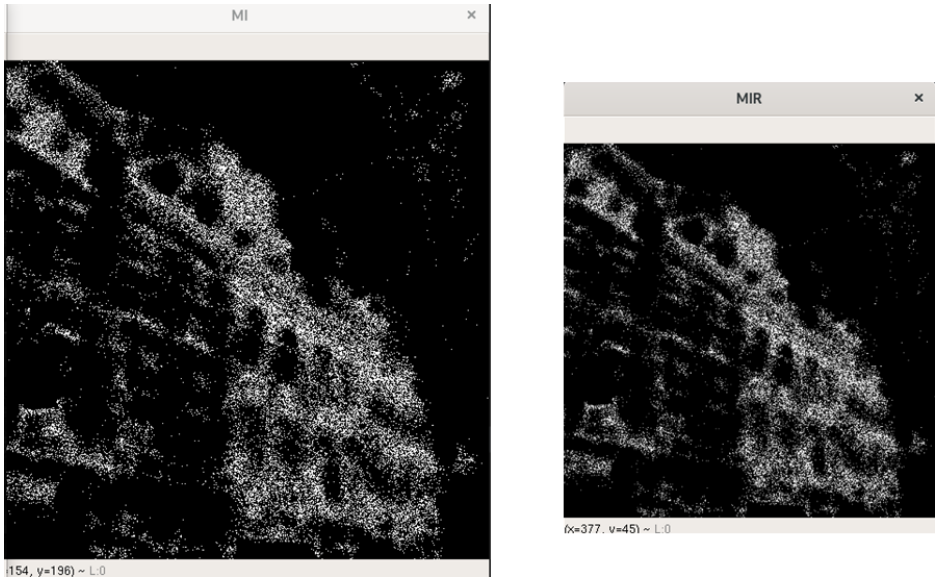


Figura 3.5. El resultado del proceso para calcular una Sub-imagen. Izquierda: a) “Mapa de inyección”. Derecha: b) “Reducción de dimensión”.

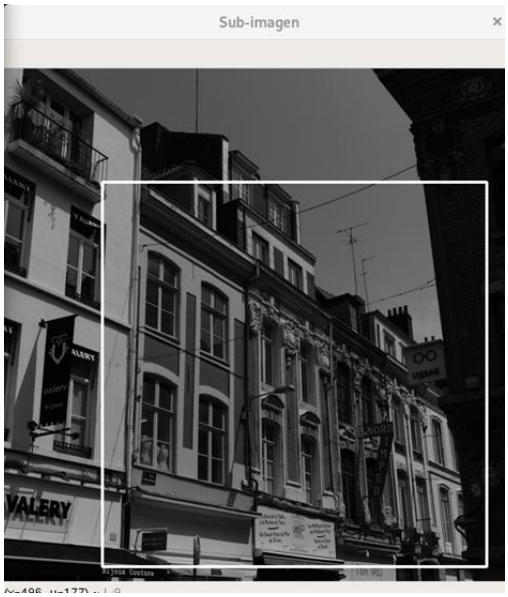


Figura 3.6. Resultado del proceso para calcular una Sub-imagen: c) “Búsqueda de la región”.

3.2.2.3 Clasificador ensamblado para estegoanálisis

El clasificador ensamblado [Kodovsky, 2011] está formado por clasificadores FLD (Discriminante Lineal de Fisher). Cada uno se entrena con una muestra *boostrap* extraída del conjunto de entrenamiento en lugar de en todo el conjunto de entrenamiento. Esta estrategia, conocida en la comunidad de aprendizaje automático como agregación *bootstrap* o *bagging* de arranque o ensacado [Breiman, 1996], también permite obtener una estimación precisa del error de prueba, que es importante para determinar los parámetros óptimos del conjunto.

La notación siguiente es para describir formalmente al clasificador ensamblado. El símbolo d representa la dimensionalidad del espacio de características, d_{sub} para la dimensionalidad del subespacio de características en el que opera cada clasificador, N^{trn} y N^{tst} son la cantidad de ejemplos de entrenamiento y prueba de cada clase, y L en el número de total de clasificadores.

Se representa, $\boldsymbol{X}_m, \overline{\boldsymbol{X}}_m \in \mathbb{R}^d, m = 1, \dots, N^{trn}$ para los vectores de características de imágenes limpias e inyectadas calculados a partir del entrenamiento y $\boldsymbol{y}_k, \overline{\boldsymbol{y}}_{km} \in \mathbb{R}^d, k = 1, \dots, N^{tst}$ para las características de prueba obtenidas de las imágenes limpias y ejemplos de inyectadas, respectivamente. Se indica el conjunto de todas las muestras de entrenamiento y prueba como $X^{trn} = \{\boldsymbol{X}_m, \overline{\boldsymbol{X}}_m\}_{m=1}^{N^{trn}}$ y $Y^{sts} = \{\boldsymbol{y}_k, \overline{\boldsymbol{y}}_k\}_{k=1}^{N^{sts}}$. Para $D \subset \{1, \dots d, \}, \boldsymbol{x}^{(D)}$ es un vector $|D|$-dimensional de características que consiste sólo en aquellas características de $\boldsymbol{x}$ cuyos índices están en D, preservando su orden original.

Los clasificadores $B_l, l = 1, \dots, L$, son asignados como $\mathbb{R}^d \rightarrow \{0,1\}$, donde 0 significa que es limpia y 1 es inyectada. El umbral de decisión del clasificador se ajusta para minimizar el error de detección total en condiciones iguales en el conjunto de entrenamiento: $P_E = \min_{P_{FA}} \frac{1}{2} (P_{FA} + P_{MD}(P_{FA}))$. Donde P_{FA}, P_{MD} son las probabilidades de falsas alarmas y detecciones perdidas, respectivamente. El pseudocódigo para el clasificador ensamblado se describe en el la Figura 3.7, y en la Figura 3.8 se muestra un diagrama conceptual.

1: **for** $l = 1$ to L **do**

2: De un subespacio aleatorio

$$\mathcal{D}_l \subset \{1, \ldots, d\}, \quad |\mathcal{D}_l| = d_{\text{sub}} < d.$$

3: De una muestra bootstrap $\mathcal{N}_l^b, \left|\mathcal{N}_l^b\right| = N^{trn}$ por muestreo uniforme con remplazo del conjunto $\{1, \ldots, N^{trn}\}$

4: Entrar un clasificador B_l con las características

$$\mathcal{X}_l = \left\{\mathbf{x}_m^{(\mathcal{D}_l)}, \bar{\mathbf{x}}_m^{(\mathcal{D}_l)}\right\}_{m \in \mathcal{N}_l^b}$$

$\Rightarrow$ obtener el eigenvector $\boldsymbol{v_l}$ y el umbral T_l

5: Para todas las muestras de ejemplo $\boldsymbol{y} \in Y^{sts}$ hacer l-esimas decisiones

$$B_l(\mathbf{y}^{(\mathcal{D}_l)}) \triangleq \begin{cases} 1, & \text{when } \mathbf{v}_l^{\mathsf{T}} \mathbf{y}^{(\mathcal{D}_l)} > T_l \\ 0, & \text{otherwise.} \end{cases}$$

6: **end for**

7: De las decisiones formar la final $B(\boldsymbol{y})$ por mayoría de votos

$$B(\mathbf{y}) = \begin{cases} 1, & \text{when } \sum_{l=1}^{L} B_l(\mathbf{y}^{(\mathcal{D}_l)}) > \frac{L}{2} \\ 0, & \text{when } \sum_{l=1}^{L} B_l(\mathbf{y}^{(\mathcal{D}_l)}) < \frac{L}{2} \\ \text{random}, & \text{otherwise.} \end{cases}$$

8: **return** $B(\mathbf{y}), \mathbf{y} \in \mathcal{Y}^{\text{tst}}$

Figura 3.7. Clasificador ensamblado [Kodovsky, 2011].

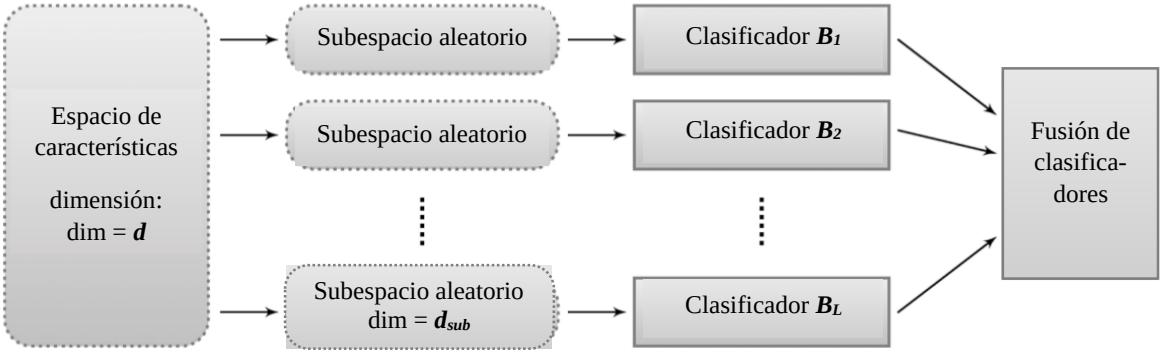


Figura 3.8. Diagrama clasificador ensamblado [Kodovsky, 2011].

3.3 Conclusiones

En este capítulo se describió la solución propuesta para el problema de la esteganografía adaptativa, al tratar dicho problema como una clasificación binaria.

A grandes rasgos, la solución propuesta se compone de dos etapas principales. La de entrenamiento y la de prueba.

Etapa de entrenamiento

En esta fase se extraen las características para entrenar al Clasificador ensamblado y crear un modelo. La principal diferencia con el enfoque típico que se ilustra en la Figura 3.1. La solución que se presentó agrega un componente antes de procesar directamente las imágenes, el cual tiene como salida la imagen en un tamaño reducido, esto se aplica a todo el conjunto de imágenes. Después se calculan las características y se entrena al clasificador.

Etapa de prueba

Es esta fase al recibir un conjunto de imágenes sin etiqueta, se extraen sus características y se continúa con su clasificación con el modelo entrenado anteriormente. Se realiza la clasificación como una imagen limpia o inyectada para el conjunto dado.

Capítulo 4

Análisis, diseño e implementación del sistema

4.1 Análisis del sistema

El problema que se planteó en este trabajo de investigación fue la clasificación de una imagen con las clase limpia o inyectada, en la práctica es un conjunto de imágenes. Conforme a la propuesta de solución presentada en el Capítulo 3, el sistema para la clasificación binaria requiere dos etapas: el entrenamiento y la predicción.

Aunado a eso, como requerimiento de los diferentes conjuntos de imágenes, se necesita que éstos sean procesados con alguno de los dos algoritmos de esteganografía que fueron seleccionaron.

Adicionalmente, se requiere que no solo sea un mensaje aleatorio embebido sino que la imagen se pueda inyectar con un mensaje de texto y que se consiga recuperar dicho mensaje.

4.2 Diseño del sistema

Cuando ya se ha analizado el problema, el paso siguiente es diseñar el sistema. Una forma de hacerlo es mediante la arquitectura del sistema, que permita integrar los requerimientos establecidos. En la siguiente sección se explica la arquitectura propuesta para este sistema.

4.2.1 Arquitectura del sistema

La arquitectura el sistema está formada principalmente por cuatro módulos, en secciones posteriores serán detallados. En la Figura 4.1 se muestra la arquitectura.

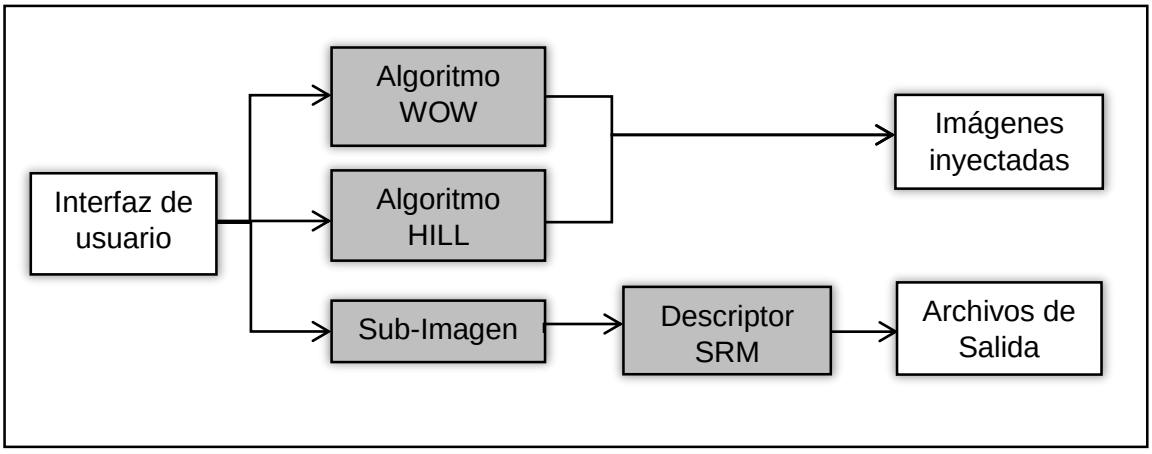


Figura 4.1. Arquitectura del sistema.

4.2.1.1 Técnicas de esteganografía

4.2.1.1.1 Función de costo HILL

Este algoritmo de esteganografía recibe un conjunto de imágenes $\boldsymbol{S_X}$ y un nivel de payload α en bits por pixel (bpp). Con estos parámetros utilizando la función de costos ρ, se genera un conjunto $\boldsymbol{S_Y}$ con la cantidad *n* de imágenes de entrada. Esto se expresa con:

$$\boldsymbol{S_X} = \{X_1, X_2, \dots X_n\} \quad (4.33)$$

donde: *n* es la cantidad de imágenes de entrada.

Para cada imagen se inyecta un mensaje *m* de longitud α con costos ρ, se genera su correspondiente inyectada.

$$\boldsymbol{Y_i} = Emb(\rho(\boldsymbol{X_i}), \boldsymbol{m}) \quad (4.34)$$

donde: $\boldsymbol{Y_i} \in \boldsymbol{S_Y} = \{Y_1, Y_2, \dots Y_n\}$.

La arquitectura para este algoritmo se muestra en la Figura 4.2.

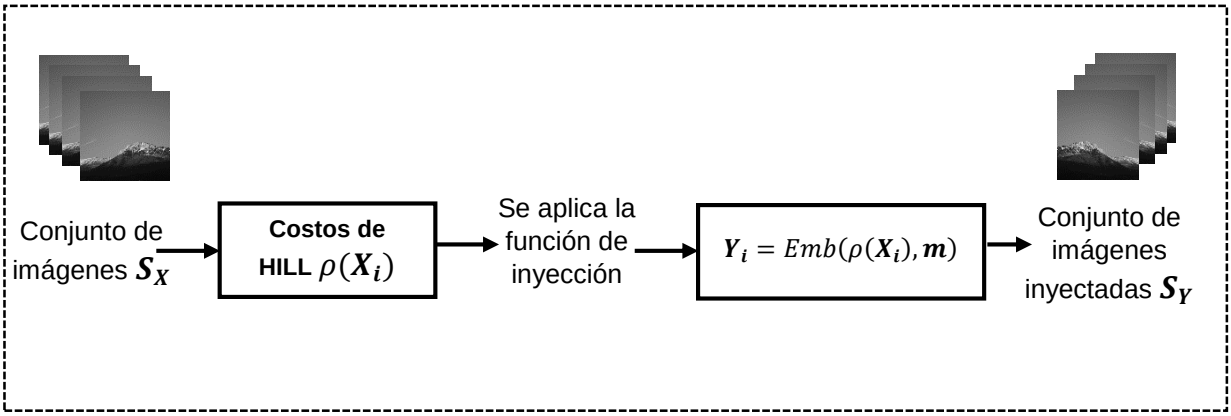


Figura 4.2. Arquitectura del algoritmo HILL.

4.2.1.1.2 Función de costo WOW

Este segundo algoritmo de esteganografía opera igual que el anterior, lo único que cambia es su función de costo. Recibe un conjunto de imágenes $\boldsymbol{S_X}$ y un

nivel de payload α en bits por pixel (bpp). Con estos parámetros utilizando la función de costos ρ, se genera un conjunto $\boldsymbol{S_Y}$ con la cantidad *n* de imágenes de entrada. Esto se expresa con:

$$\boldsymbol{S_X} = \{X_1, X_2, \dots X_n\} \quad (4.35)$$

donde: *n* es la cantidad de imágenes de entrada.

Para cada imagen se inyecta un mensaje *m* de longitud α con costos ρ, se genera su correspondiente inyectada.

$$\boldsymbol{Y_i} = Emb(\rho(\boldsymbol{X_i}), \boldsymbol{m}) \quad (4.36)$$

donde: $\boldsymbol{Y_i} \in \boldsymbol{S_Y} = \{Y_1, Y_2, \dots Y_n\}$.

La arquitectura para este algoritmo es la Figura 4.3

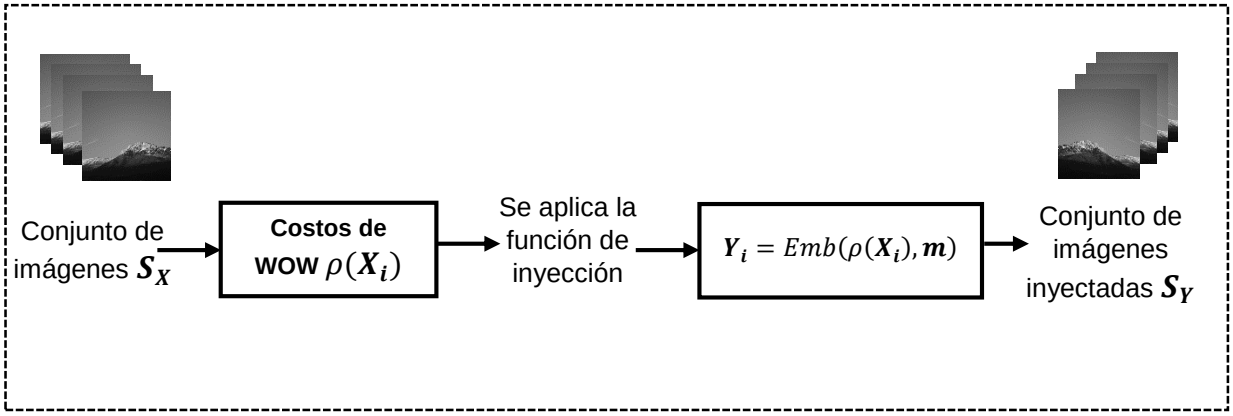


Figura 4.3. Arquitectura del algoritmo WOW.

4.2.1.2 Técnicas de estegoanálisis

4.2.1.2.1 Descriptor SRM

Dado un conjunto de imágenes $\boldsymbol{S}$, se calcula para cada una de ellas su correspondiente vector de características $\boldsymbol{v_i} = \{r_1, r_2, \dots r_n\}$, donde n = 34,671, y r_i es el residuo de ruido calculado. Después se guardan los vectores de todas las imágenes en un archivo. La arquitectura para este descriptor global se muestra en la Figura 4.4.

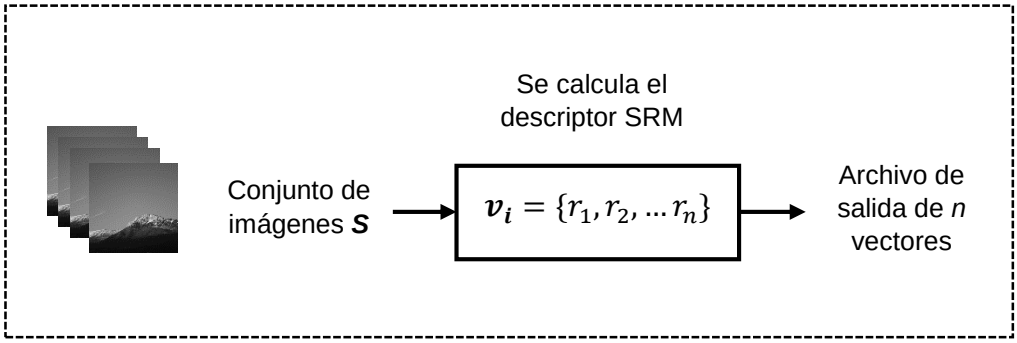


Figura 4.4. Arquitectura del descriptor SRM.

4.2.1.2.2 Proceso para generar Sub-imágenes

Dado un conjunto de imágenes $\boldsymbol{S} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n\}$, donde *n* es el número total de imágenes, se calcula para cada una de éstas una Sub-imagen. Como parámetro se utiliza $tR,\ 0 < tR < 1$, que es la tamaño de reducción. Se genera un conjunto ***R*** de ***n*** elementos. El proceso para el cálculo de la Sub-imagen se ilustra en la siguiente Figura 4.5.

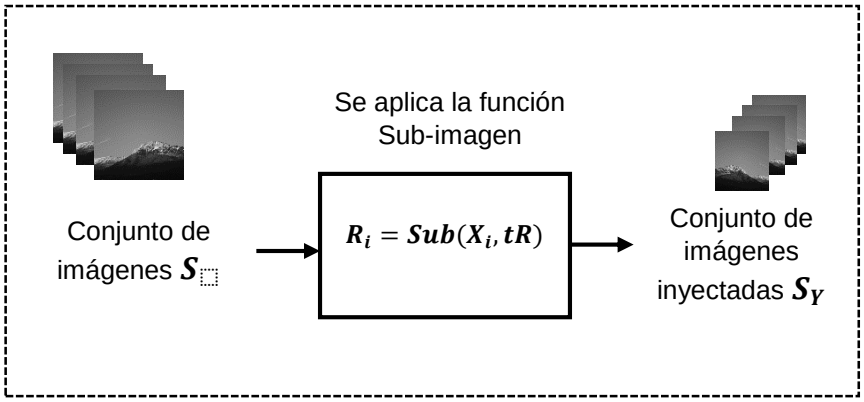


Figura 4.5. Arquitectura del proceso Sub-imagen.

4.3 Implementación del sistema

El sistema fue implementado en el lenguaje Python con apoyo del IDE PyCharm. Algunas implementaciones fueron agregadas directamente de funciones en Matlab desarrolladas por sus correspondientes autores, que están disponibles

en (http://dde.binghamton.edu/download/). Estas funciones en Matlab permiten ser llamadas desde Python. Se adaptaron técnicas para optimizar el rendimiento del hardware mediante subprocesos en el lenguaje Python.

4.3.1 Interfaz de usuario

Para la interfaz del usuario se presenta la pantalla principal. Se utilizó la librería TKinter para crear la interfaz del usuario. En la Figura 4.6 se presenta la interfaz gráfica de usuario (GUI) del sistema.

Figura 4.6. GUI del Sistema, Pantalla principal.

Como se puede observar la interfaz cuenta con un menú con las dos partes principales de esta investigación, la estenografía y su contramedida el estegoanálisis. Para el caso de la esteganografía, el algoritmo HILL tiene la funcionalidad para incrustar un mensaje de texto proveniente del usuario. En el apartado “Opciones” el usuario puede configurar la ruta de los archivos de salida. Así como, variar el payload al procesar las imágenes con un método de esteganografía.

4.3.2 Archivos de entrada y salida

Estos archivos procesados son muy importantes, su correcta estructura permite que el sistema funcione correctamente. Por el contrario si no se reconoce la estructura del archivo el sistema no podrá realizar acciones. Los archivos de entrada y salida se detallan a continuación.

Archivos de entrada

- Para la esteganografía las imágenes del banco BOSSbase están en formato PGM RAW o bruto, si bien no solo se puede procesar este formato. El formato que no se puede procesar son las imágenes JPEG / JPG.
- En la inyección del mensaje de texto para el algoritmo HILL, es preferible usar archivos con extensión .txt, debido a se puede leer de forma errónea el mensaje que se pretende ocultar cuando se examinen los bits de la cabecera del archivo.

Archivos de salida

- Los archivos de salida se crean cuando se utiliza el método de la Sub-imagen, son guardados directamente en el mismo formato PGM sin compresión y sin pérdida.
- Los resultados de aplicar el descriptor SRM, se guardan en dos archivos, uno de ellos es el que contendrá los vectores de características y su extensión es *.fea*, en el segundo archivo se guardan los nombres de las imágenes que fueron procesadas en el orden en que se aplicó el descriptor, su extensión es *.label*.

4.3.3 Lenguajes de programación

Como ya se mencionó este sistema fue desarrollado usando el lenguaje Pyhton, y se trabajó en el sistema operativo GNU/Linux Debian 10 Booster. Al instalar las librerías necesarias, se prevé que tenga el mismo funcionamiento en

sistemas Microsoft Windows, como beneficio que el lenguaje Python es interpretado y no compilado.

En el sistema desarrollado otros lenguajes son necesarios, por ejemplo el leguaje de Matlab, para utilizar las características con el clasificador ensamblado, el cual está implementado en dicho lenguaje. Asimismo, el lenguaje GNU Octave es utilizado para calcular las características de forma más rápida, con subprocesos desde el lenguaje Pyhton.

4.4 Conclusiones

En este capítulo se describió el análisis, diseño, e implementación del sistema. Fue presentada la arquitectura del sistema en la Sección 4.2.1. Este software fue construido de forma modular lo que permite que pueda ser reutilizado en futuras investigaciones.

El sistema cuenta con dos etapas principales, planteadas en capítulo 3, la etapa de entrenamiento y la de clasificación de una imagen. La arquitectura de éstas fue descrita en la sección 4.2.1, junto con los algoritmos de incrustación.

Capítulo 5

Pruebas y resultados

5.1 Ambiente de pruebas

En este apartado se muestran las características del equipo de cómputo que se utilizó para realizar todas las pruebas. También, en la siguiente sección se describe el banco de imágenes utilizado. El sistema de cómputo que se utilizó tiene las siguientes características.

- Sistema Operativo Debian GNU/Linux 10 Buster
- Procesador Intel Core i7-9700K 4.6GHz x 8
- 16 GB de memoria RAM / 2GB de intercambio (Swap)
- Unidad de estado sólido 480GB

5.2 Banco de imágenes

El banco de imágenes con el que se trabajó fue BOSSBase v1.01 [Bas, 2011], el cual contiene 10,000 imágenes limpias en formato bruto PGM, cada imagen tiene un tamaño de 512x512, y están en escala de grises.

5.3 Plan de pruebas

Las pruebas realizadas se constituyen de tres etapas principales. Se trata de detectar los dos algoritmos de esteganografía que fueron seleccionados. Como primera etapa se buscaron los umbrales para el algoritmo que se propuso, descrito en la Sección 3.2.2.2 “Proceso para generar Sub-imágenes”. Las dos etapas restantes fueron: probar el enfoque propuesto, y probar el enfoque de la literatura.

5.3.1 Descripción de las pruebas

A. Obtención de umbrales para el algoritmo propuesto

El objetivo de esta prueba fue encontrar los umbrales adecuados para el algoritmo propuesto. Como se describió en la Sección 3.2.2.2, se tienen tres umbrales, el primero es el tamaño de reducción del mapa de inyección, este valor se multiplica por el tamaño de la imagen, obteniendo de esta forma un tamaño reducido. Un segundo umbral es el tamaño de exploración en el mapa de inyección, que es el que busca la parte que tiene mayor inyección de un tamaño dado. Como último umbral es el salto al ir procesando la imagen, cada vez que se calcule la parte de mayor inyección en la imagen.

Para este caso en específico se utilizaron 4,000 imágenes limpias del banco de imágenes BOSSBase v1.01 y 4,000 imágenes inyectadas con el algoritmo HILL con 0.40 bpp.

B. Detección con el método propuesto

El objetivo de esta prueba es examinar el rendimiento del enfoque propuesto para detectar los dos algoritmos de esteganografía que se seleccionaron. Primero se generan los conjuntos de sub-imágenes limpias e inyectas, se entrena el modelo

y se prueba el modelo para determinar cuáles imágenes están limpias y cuáles tienen un mensaje oculto, de las imágenes que no se utilizaron para el entrenamiento. Para entrenar el modelo se utilizó el clasificador ensamblado.

En este caso se prueba con imágenes de tamaño 512x512 el cual se abrevia como SubSRM, y se prueba con sub-imágenes de tamaño 384x384 a esta última se le nombró como Sub2SRM. Los algoritmos son: el algoritmo HILL y el algoritmo WOW, con tres niveles de carga, con 0.30 bpp (bits por pixel), con 0.40 bpp y con 0.50 bpp.

C. Detección con el SRM

El objetivo de esta prueba es replicar el rendimiento del enfoque del Modelo enriquecido espacial para detectar los dos algoritmos de esteganografía que fueron seleccionados. Se crean los conjuntos de imágenes limpias e inyectadas, se entrenó el modelo y éste se para determinar cuáles imágenes están limpias y cuáles tienen un mensaje oculto. Se utilizó el clasificador ensamblado para entrenar el modelo. De igual forma, se realizó el mismo proceso para tres niveles de carga con el algoritmo HILL y el algoritmo WOW, con 0.03 bpp, 0.40 bpp y con 0.50 bpp.

5.3.2 Esteganografía

A excepción de la primer prueba de la sección 5.3.3.1, en todas las pruebas se utilizaron los algoritmos de esteganografía *High-pass, Low-pass and Low-pass* y *Wavelet obteined Weings* con Payloads de 0.30 bits por pixel, 0.40 bits por pixel y 0.50 bits por pixel. El mensaje inyectado fue un mensaje aleatorio, al realizar la inyección de acuerdo a cada algoritmo se modificaron los pixeles a $y_{ij} = \{x_{ij} - 1, x_{ij}, x_{ij} + 1\}$, en un rango de 0 a 255 en el valor del pixel.

5.3.3 Estegoanálisis

A continuación se muestran todas las pruebas realizadas con los diferentes enfoques.

5.3.3.1 Obtención de umbrales

En este apartado se calcularon los umbrales para el método planteado. Como se detalla en la Sección 3.2.2.2, el proceso tiene varios parámetros los cuales se encontraron con la siguiente experimentación. Los parámetros se detallan a continuación.

a) *tR* es un umbral de reducción del mapa de inyección y también indica el tamaño de la sub-imagen generada.
b) Tamaño de la exploración para calcular el área, se dejó en la mitad de las dimensiones de *tR*.
c) *Salto* al realizar la búsqueda de la sub-imagen, se realiza sumando "salto" a los índices *ij.*

Para estos experimentos se utilizó el *framework* Aletheia, que tiene funciones implementadas para esteganografía y estegoanálisis, está disponible en *(https://github.com/daniellerch/aletheia)*. Se inyectaron 4,000 imágenes con el algoritmo HILL con 0.40 bpp, y se utilizó el clasificador ensamblado para el entrenamiento. La implementación del *framework* Aletheia permite calcular la exactitud por defecto, está implementado para dividir el conjunto para entrenamiento en 90% y 10% restante para prueba. Ésta división se puede modificar pero sólo a nivel de código. Los resultados se despliegan en la Tabla 5.1.

Tabla 5.1.Experimentos con diferentes umbrales de la sub-imagen.

No.	Tamaño Sub-imagen	Exactitud
1	256x256	0.68%
2	128x128	0.64%
3	384x384	0.70%
4*	512x512	0.72%

En la Tabla 5.1 se observa que el mejor rendimiento para las diferentes sub-imágenes fue con un tamaño de 384x384. También se muestra el experimento 4,

en el que no se implementa la sub-imagen (tamaño normal del banco de imágenes 512x512). Los parámetros de tamaño en 384x384, fueron los siguientes. El parámetro *tR* se calculó incrementándolo con relación al nivel de detección.

a) *tR*, tamaño de reducción del mapa de inyección: 0.75.
b) Tamaño de la exploración para calcular el área: 192x192.
c) *Salto* al realizar la búsqueda: 10.

Con estos parámetros se realizaron las pruebas mostradas en las próximas secciones, se evalúa el enfoque y se compara con lo reportado en la literatura.

5.3.3.2 Detección con el algoritmo propuesto

Se reporta el resultado del enfoque propuesto, con diez divisiones aleatorias de 50% del conjunto de imágenes para entrenamiento y el 50% restante para pruebas. Como métricas de evaluación se utilizó el error de detección, el *Accuracy* y *F-measure*. Se calculan con las siguientes ecuaciones.

$$P_E = \min_{P_{FA}} \frac{1}{2} \left(P_{FA} + P_{MD}(P_{FA})\right) \quad (5.37)$$

Donde: P_{FA} son las probabilidades de falsa alarma (FP: Falsos positivos) y P_{MD} las detecciones perdidas (FN: Falsos negativos).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.38)$$

Donde. TP: Verdaderos Positivos, TN: Verdaderos Negativos, FP: Falsos Positivos y FN: Falsos Negativos.

$$F - measure = \frac{2TP}{2TP + FP + FN} \quad (5.39)$$

Donde. TP: Verdaderos Positivos, FP: Falsos Positivos y FN: Falsos Negativos.

La matriz de confusión que se reporta en las próximas secciones se muestra en la Tabla 5.2.

Tabla 5.2. Matriz de confusión.

		Clase Real	
		Stego	*Cover*
Clase Predicha	*Stego*	TP	FP
	Cover	FN	TN

5.3.3.2.1 Entrenamiento 1 y 2 con payload 0.30 bpp

Algoritmo HILL

Prueba SubSRM. Para este caso se realizó el entrenamiento con el método propuesto y se probó con imágenes de tamaño 512x512 con el algoritmo HILL. En la Tabla 5.3 se muestra el número de clasificadores utilizados. En la Tabla 5.4 se detallan las matrices de confusión con el resultado de las pruebas con el 50% del conjunto de imágenes que no se utilizaron en el entrenamiento. Los resultados de las métricas se exponen en la Tabla 5.5.

Tabla 5.3. SubSRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	106	102	135	101	105	127	117	87	99	101
Tiempo total (HH:MM:SS)	00:14:22									

Tabla 5.4. SubSRM, detección de HILL con Payload 0.3 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	2883	1079	**S**	2921	1108	**S**	2943	1078	**S**	2811	1107	**S**	2925	1155
C	2117	3921	**C**	2079	3892	**C**	2057	3922	**C**	2189	3893	**C**	2075	3845
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	2878	1119	**S**	2983	1145	**S**	2927	1204	**S**	2937	1117	**S**	2859	1128
C	2122	3881	**C**	2017	3855	**C**	2073	3796	**C**	2063	3883	**C**	2141	3872
Tiempo total	00:00:59													

Tabla 5.5. SubSRM, rendimiento de modelos con HILL.

Prueba	P_E	Accuracy	F-Measure
1	0.3196	0.6804	0.6434
2	0.3187	0.6813	0.6470
3	0.3135	0.6865	0.6525
4	0.3296	0.6704	0.6304
5	0.3230	0.6770	0.6443
6	0.3241	0.6759	0.6398
7	0.3162	0.6838	0.6536
8	0.3277	0.6723	0.6411
9	0.3180	0.6820	0.6488
10	0.3269	0.6731	0.6363
Media	**0.3217**	**0.6783**	**0.6437**

Prueba Sub2SRM. En esta prueba se utilizó el entrenamiento generado anteriormente, solo cambio al probarlo con imágenes de tamaño 384x384 para el algoritmo HILL. Las matrices de confusión con el algoritmo HILL se exponen en la Tabla 5.6. Las métricas se muestran en la Tabla 5.7.

Tabla 5.6. Sub2SRM, detección de HILL con Payload 0.3 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3471	1551	**S**	3476	1581	**S**	3398	1499	**S**	3428	1535	**S**	3500	1583
C	1529	3449	**C**	1524	3419	**C**	1602	3501	**C**	1572	3465	**C**	1500	3417
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3448	1541	**S**	3517	1593	**S**	3547	1670	**S**	3513	1573	**S**	3498	1610
C	1552	3459	**C**	1483	3407	**C**	1453	3330	**C**	1487	3427	**C**	1502	3390
Tiempo total			00:00:45											

Tabla 5.7. Sub2SRM, rendimiento de modelos con HILL.

Prueba	P_E	Accuracy	F-Measure
1	0.3080	0.6920	0.6927
2	0.3105	0.6895	0.6913
3	0.3101	0.6899	0.6867
4	0.3107	0.6893	0.6881
5	0.3083	0.6917	0.6942
6	0.3093	0.6907	0.6904
7	0.3076	0.6924	0.6957
8	0.3123	0.6877	0.6943
9	0.3060	0.6940	0.6966
10	0.3112	0.6888	0.6921
Media	**0.3094**	**0.6906**	**0.6922**

Algoritmo WOW

Prueba SubSRM. En este caso se realizó el entrenamiento con el método propuesto y se probó con imágenes de tamaño 512x512 para detectar el algoritmo WOW. En la Tabla 5.8 se muestra el número de clasificadores utilizados. En la Tabla 5.9 se detallan las matrices de confusión. Los resultados de las métricas se exponen en la Tabla 5.10.

Tabla 5.8. SubSRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	115	102	93	91	107	106	109	103	123	118
Tiempo total (HH:MM:SS)	00:11:02									

Tabla 5.9. SubSRM, detección de WOW con 0.30 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3145	995	**S**	3109	956	**S**	3108	990	**S**	3042	1011	**S**	3118	932
C	1855	4005	**C**	1891	4044	**C**	1892	4010	**C**	1958	3989	**C**	1882	4068

Tabla 5.9. Continuación.

***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3119	947	**S**	3130	1016	**S**	3129	974	**S**	3172	996	**S**	3126	975
C	1881	4053	**C**	1870	3984	**C**	1871	4026	**C**	1828	4004	**C**	1874	4025
Tiempo total			00:00:47											

Tabla 5.10. SubSRM, rendimiento de modelos con WOW.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2850	0.7150	0.6882
2	0.2847	0.7153	0.6859
3	0.2882	0.7118	0.6832
4	0.2969	0.7031	0.6720
5	0.2814	0.7186	0.6891
6	0.2828	0.7172	0.6881
7	0.2886	0.7114	0.6845
8	0.2845	0.7155	0.6875
9	0.2824	0.7176	0.6920
10	0.2849	0.7151	0.6870
Media	**0.2859**	**0.7141**	**0.6857**

Prueba Sub2SRM. En este caso se utilizó el entrenamiento anterior con el método propuesto, y se probó con imágenes de tamaño 384x384 para detectar el algoritmo WOW. En la Tabla 5.11 se detallan las matrices de confusión. Los resultados de las métricas se exponen en la Tabla 5.12.

Tabla 5.11. Sub2SRM, detección de WOW con 0.30 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3711	1409	**S**	3722	1418	**S**	3737	1417	**S**	3672	1452	**S**	3686	1335
C	1289	3591	**C**	1278	3582	**C**	1263	3583	**C**	1328	3548	**C**	1314	3665

Tabla 5.11. Continuación.

Split 6			*Split* 7			*Split* 8			*Split* 9			*Split* 10		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3665	1365	**S**	3752	1454	**S**	3694	1387	**S**	3776	1400	**S**	3757	1408
C	1335	3635	**C**	1248	3546	**C**	1306	3613	**C**	1224	3600	**C**	1243	3592
Tiempo total			00:00:52											

Tabla 5.12. Sub2SRM, rendimiento de modelos con WOW.

Prueba	P_E	Accuracy	F-Measure
1	0.2698	0.7302	0.7334
2	0.2696	0.7304	0.7341
3	0.2680	0.7320	0.7361
4	0.2780	0.7220	0.7254
5	0.2649	0.7351	0.7357
6	0.2700	0.7300	0.7308
7	0.2702	0.7298	0.7353
8	0.2693	0.7307	0.7329
9	0.2624	0.7376	0.7421
10	0.2651	0.7349	0.7392
Media	**0.2687**	**0.7313**	**0.7345**

5.3.3.2.2 Entrenamiento 3 y 4 con payload 0.40 bpp

Algoritmo HILL

Prueba SubSRM. Para este caso se realizó el entrenamiento con el método propuesto con 50% del conjunto de imágenes y se probó con 50% de imágenes que no se utilizaron en el entrenamiento de tamaño 512x512 con el algoritmo HILL. En la Tabla 5.13 se muestra el número de clasificadores utilizados. En la Tabla 5.14 se detallan las matrices de confusión con el resultado de las pruebas. Los resultados de las métricas se exponen en la Tabla 5.15.

Tabla 5.13. SubSRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	114	105	74	101	95	112	111	142	88	95
Tiempo total (HH:MM:SS)	00:09:36									

Tabla 5.14. SubSRM, detección de HILL con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3200	945	**S**	3195	921	**S**	3218	1059	**S**	3161	903	**S**	3177	921
C	1800	4055	**C**	1805	4079	**C**	1782	3941	**C**	1839	4097	**C**	1823	4079
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3126	922	**S**	3203	964	**S**	3164	917	**S**	3188	981	**S**	3164	943
C	1874	4078	**C**	1797	4036	**C**	1836	4083	**C**	1812	4019	**C**	1836	4057
Tiempo total	00:00:40													

Tabla 5.15. SubSRM, rendimiento de modelos con HILL.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2745	0.7255	0.6998
2	0.2726	0.7274	0.7010
3	0.2841	0.7159	0.6938
4	0.2742	0.7258	0.6975
5	0.2744	0.7256	0.6984
6	0.2796	0.7204	0.6910
7	0.2761	0.7239	0.6988
8	0.2753	0.7247	0.6968
9	0.2793	0.7207	0.6954
10	0.2779	0.7221	0.6949
Media	**0.2768**	**0.7232**	**0.6967**

Prueba Sub2SRM. En esta prueba se utilizó el entrenamiento anterior con el método propuesto, y se probó con imágenes de tamaño 384x384 para detectar el

algoritmo HILL. En la Tabla 5.16 se detallan las matrices de confusión. Los resultados de las métricas se muestran en la Tabla 5.17.

Tabla 5.16. Sub2SRM, detección de HILL con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3776	1368	**S**	3758	1376	**S**	3851	1509	**S**	3764	1315	**S**	3761	1326
C	1224	3632	**C**	1242	3624	**C**	1149	3491	**C**	1236	3685	**C**	1239	3674
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3766	1354	**S**	3811	1409	**S**	3777	1350	**S**	3848	1425	**S**	3807	1361
C	1234	3646	**C**	1189	3591	**C**	1223	3650	**C**	1152	3575	**C**	1193	3639
Tiempo total			00:00:39											

Tabla 5.17. Sub2SRM, rendimiento de modelos con HILL.

Prueba	P_E	Accuracy	F-Measure
1	0.2592	0.7408	0.7445
2	0.2618	0.7382	0.7417
3	0.2658	0.7342	0.7434
4	0.2551	0.7449	0.7469
5	0.2565	0.7435	0.7457
6	0.2588	0.7412	0.7443
7	0.2598	0.7402	0.7458
8	0.2573	0.7427	0.7459
9	0.2577	0.7423	0.7491
10	0.2554	0.7446	0.7488
Media	**0.2587**	**0.7413**	**0.7456**

Algoritmo WOW

Prueba SubSRM. En este caso se realizó el entrenamiento con el método propuesto y se probó con imágenes de tamaño 512x512 para detectar el algoritmo WOW. En la Tabla 5.18 se muestra el número de clasificadores utilizados. En la Tabla 5.19 se detallan las matrices de confusión. Los resultados de las métricas se exponen en la Tabla 5.20.

Tabla 5.18. SubSRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	97	116	109	106	117	87	89	106	90	107
Tiempo total (HH:MM:SS)	00:13:31									

Tabla 5.19. SubSRM, detección de WOW con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3470	800	**S**	3410	740	**S**	3454	715	**S**	3357	774	**S**	3449	772
C	1530	4200	**C**	1590	4260	**C**	1546	4285	**C**	1643	4226	**C**	1551	4228
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3358	806	**S**	3404	755	**S**	3385	750	**S**	3376	810	**S**	3415	760
C	1642	4194	**C**	1596	4245	**C**	1615	4250	**C**	1624	4190	**C**	1585	4240
Tiempo total	00:00:48													

Tabla 5.20. SubSRM, rendimiento de modelos con WOW.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2330	0.7670	0.7487
2	0.2330	0.7670	0.7454
3	0.2261	0.7739	0.7534
4	0.2417	0.7583	0.7353
5	0.2323	0.7677	0.7481
6	0.2448	0.7552	0.7329
7	0.2351	0.7649	0.7433
8	0.2365	0.7635	0.7411
9	0.2434	0.7566	0.7350
10	0.2345	0.7655	0.7444
Media	**0.2360**	**0.7640**	**0.7428**

Prueba Sub2SRM. En esta prueba se utilizó el entrenamiento anterior con el método propuesto, y se probó con imágenes de tamaño 384x384 para detectar el

algoritmo WOW. En las Tabla 5.21 y 5.22 se detallan las matrices de confusión y los resultados de las métricas, respectivamente.

Tabla 5.21. Sub2SRM, detección de WOW con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3989	1188	**S**	3953	1146	**S**	3908	1098	**S**	3936	1202	**S**	3970	1151
C	1011	3812	**C**	1047	3854	**C**	1092	3902	**C**	1064	3798	**C**	1030	3849
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3935	1223	**S**	3954	1183	**S**	3928	1123	**S**	3953	1208	**S**	4003	1184
C	1065	3777	**C**	1046	3817	**C**	1072	3877	**C**	1047	3792	**C**	997	3816
Tiempo total			00:00:46											

Tabla 5.22. Sub2SRM, rendimiento de modelos con WOW.

Prueba	P_E	Accuracy	F-Measure
1	0.2199	0.7801	0.7839
2	0.2193	0.7807	0.7828
3	0.2190	0.7810	0.7811
4	0.2266	0.7734	0.7765
5	0.2181	0.7819	0.7845
6	0.2288	0.7712	0.7748
7	0.2229	0.7771	0.7801
8	0.2195	0.7805	0.7816
9	0.2255	0.7745	0.7781
10	0.2181	0.7819	0.7859
Media	**0.2218**	**0.7782**	**0.7809**

5.3.3.2.3 Entrenamiento 5 y 6 con payload 0.50 bpp

Algoritmo HILL

Prueba SubSRM. Para este caso se realizó el entrenamiento con el método propuesto con 50% del conjunto de imágenes y se probó con 50% de imágenes que no se utilizaron en el entrenamiento, de tamaño 512x512 con el algoritmo HILL. En la Tabla 5.23 se muestra el número de clasificadores utilizados. En las Tabla 5.24 y

5.25 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.23. SubSRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	98	96	86	91	95	108	86	78	123	109
Tiempo total (HH:MM:SS)	00:08:05									

Tabla 5.24. SubSRM, detección de HILL con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3393	751	**S**	3448	750	**S**	3435	775	**S**	3396	756	**S**	3420	743
C	1607	4249	**C**	1552	4250	**C**	1565	4225	**C**	1604	4244	**C**	1580	4257
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3421	777	**S**	3469	821	**S**	3416	753	**S**	3447	733	**S**	3490	751
C	1579	4223	**C**	1531	4179	**C**	1584	4247	**C**	1553	4267	**C**	1510	4249
Tiempo total			00:00:45											

Tabla 5.25. SubSRM, rendimiento de modelos con HILL.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2358	0.7642	0.7421
2	0.2302	0.7698	0.7497
3	0.2340	0.7660	0.7459
4	0.2360	0.7640	0.7421
5	0.2323	0.7677	0.7465
6	0.2356	0.7644	0.7439
7	0.2352	0.7648	0.7468
8	0.2337	0.7663	0.7451
9	0.2286	0.7714	0.7510
10	0.2261	0.7739	0.7553
Media	**0.2328**	**0.7672**	**0.7469**

Prueba Sub2SRM. En esta prueba se utilizó el entrenamiento anterior con el método propuesto, y se probó con imágenes de tamaño 384x384 para detectar el algoritmo HILL. En las Tabla 5.26 y 5.27 se muestran las matrices de confusión y los resultados de las métricas, respectivamente.

Tabla 5.26. Sub2SRM, detección de HILL con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3949	1149	**S**	4009	1137	**S**	3988	1173	**S**	3957	1158	**S**	3985	1128
C	1051	3851	**C**	991	3863	**C**	1012	3827	**C**	1043	3842	**C**	1015	3872
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3985	1174	**S**	4062	1248	**S**	3969	1169	**S**	3984	1136	**S**	4005	1146
C	1015	3826	**C**	938	3752	**C**	1031	3831	**C**	1016	3864	**C**	995	3854
Tiempo total			00:00:46											

Tabla 5.27. Sub2SRM, rendimiento de modelos con HILL.

Prueba	P_E	Accuracy	F-Measure
1	0.2200	0.7800	0.7821
2	0.2128	0.7872	0.7903
3	0.2185	0.7815	0.7850
4	0.2201	0.7799	0.7824
5	0.2143	0.7857	0.7881
6	0.2189	0.7811	0.7845
7	0.2186	0.7814	0.7880
8	0.2200	0.7800	0.7830
9	0.2152	0.7848	0.7874
10	0.2141	0.7859	0.7891
Media	**0.2172**	**0.7827**	**0.7860**

Algoritmo WOW

Prueba SubSRM. En este caso se efectuó el entrenamiento con el método propuesto y se probó con imágenes de tamaño 512x512 para detectar el algoritmo WOW. En la Tabla 5.28 se muestra el número de clasificadores utilizados. En las

Tabla 5.29 y 5.30 se detallan las matrices de confusión y los resultados de las métricas, respectivamente.

Tabla 5.28. SubSRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	109	85	121	109	111	132	103	119	102	87
Tiempo total (HH:MM:SS)	00:15:27									

Tabla 5.29. SubSRM, detección de WOW con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3638	664	**S**	3653	618	**S**	3709	617	**S**	3714	623	**S**	3638	608
C	1362	4336	**C**	1347	4382	**C**	1291	4383	**C**	1286	4377	**C**	1362	4392
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3664	596	**S**	3659	629	**S**	3697	603	**S**	3667	613	**S**	3640	652
C	1336	4404	**C**	1341	4371	**C**	1303	4397	**C**	1333	4387	**C**	1360	4348
Tiempo total			00:00:57											

Tabla 5.30. SubSRM, rendimiento de modelos con WOW.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2026	0.7974	0.7822
2	0.1965	0.8035	0.7880
3	0.1908	0.8092	0.7954
4	0.1909	0.8091	0.7955
5	0.1970	0.8030	0.7869
6	0.1932	0.8068	0.7914
7	0.1970	0.8030	0.7879
8	0.1906	0.8094	0.7951
9	0.1946	0.8054	0.7903
10	0.2012	0.7988	0.7835
Media	**0.1954**	**0.8046**	**0.7896**

Prueba Sub2SRM. En esta prueba se utilizó el entrenamiento anterior con el método propuesto, y se probó con imágenes de tamaño 384x384 para detectar el algoritmo WOW. En las Tabla 5.31 y 5.32 se muestran las matrices de confusión y los resultados de las métricas, respectivamente.

Tabla 5.31. Sub2SRM, detección de WOW con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4151	1063	**S**	4141	971	**S**	4170	933	**S**	4149	986	**S**	4124	941
C	849	3937	**C**	859	4029	**C**	830	4067	**C**	851	4014	**C**	876	4059
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4095	944	**S**	4151	974	**S**	4148	952	**S**	4132	951	**S**	4179	1046
C	905	4056	**C**	849	4026	**C**	852	4048	**C**	868	4049	**C**	821	3954
Tiempo total			00:00:57											

Tabla 5.32. Sub2SRM, rendimiento de modelos con WOW.

Prueba	P_E	Accuracy	F-Measure
1	0.1912	0.8088	0.8128
2	0.1830	0.8170	0.8190
3	0.1763	0.8237	0.8255
4	0.1837	0.8163	0.8187
5	0.1817	0.8183	0.8195
6	0.1849	0.8151	0.8158
7	0.1823	0.8177	0.8200
8	0.1804	0.8196	0.8214
9	0.1819	0.8181	0.8196
10	0.1867	0.8133	0.8174
Media	**0.1832**	**0.8168**	**0.8190**

5.3.3.3 Detección con SRM

Esta prueba tiene como objetivo replicar el rendimiento del enfoque del *Modelo Enriquecido Espacial* para detectar los dos algoritmos de esteganografía que fueron seleccionados. En secciones posteriores se analiza su comportamiento respecto al rendimiento del enfoque propuesto.

Se crean los conjuntos de imágenes limpias e inyectas, se entrenó el modelo y se prueba para determinar cuáles imágenes están limpias y cuáles están inyectadas. Se utilizó el clasificador ensamblado para entrenar el modelo. Se realizó el mismo proceso para tres niveles de carga con el algoritmo HILL y el algoritmo WOW, con 0.03 bpp, 0.40 bpp y con 0.50 bpp.

Se reporta el resultado con diez divisiones aleatorias de 50% del conjunto de imágenes para entrenamiento y el 50% restante para pruebas. Las métricas utilizadas fueron las antes descritas, el error de detección, el *Accuracy* y *F-measure*.

5.3.3.3.1 Entrenamiento 7 y 8 con payload 0.30 bpp

Algoritmo HILL

En este caso se realizó el entrenamiento con el SRM para detectar el algoritmo HILL. En la Tabla 5.33 se muestra el número de clasificadores utilizados. En las Tabla 5.34 y 5.35 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.33. SRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	115	109	111	83	94	116	94	115	99	87
Tiempo total (HH:MM:SS)	00:10:34									

Tabla 5.34. SRM, detección de HILL con 0.30 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3536	1538	**S**	3510	1510	**S**	3494	1446	**S**	3512	1523	**S**	3580	1640
C	1464	3462	**C**	1490	3490	**C**	1506	3554	**C**	1488	3477	**C**	1420	3360
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3462	1430	**S**	3546	1541	**S**	3540	1528	**S**	3583	1577	**S**	3554	1497
C	1538	3570	**C**	1454	3459	**C**	1460	3472	**C**	1417	3423	**C**	1446	3503
Tiempo total			00:00:48											

Tabla 5.35. SRM, rendimiento de modelos con HILL.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.3002	0.6998	0.7020
2	0.3000	0.7000	0.7006
3	0.2952	0.7048	0.7030
4	0.3011	0.6989	0.7000
5	0.3060	0.6940	0.7006
6	0.2968	0.7032	0.7000
7	0.2995	0.7005	0.7031
8	0.2988	0.7012	0.7032
9	0.2994	0.7006	0.7053
10	0.2943	0.7057	0.7072
Media	**0.2991**	**0.7009**	**0.7025**

Algoritmo WOW

En esta prueba se realizó el entrenamiento con el SRM para detectar el algoritmo WOW. En la Tabla 5.36 se muestra el número de clasificadores utilizados. En las Tabla 5.37 y 5.38 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.36. SRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	96	117	100	92	95	131	124	93	100	110
Tiempo total (HH:MM:SS)	00:18:36									

Tabla 5.37. SRM, detección de WOW con 0.30 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3771	1373	**S**	3735	1349	**S**	3735	1308	**S**	3723	1282	**S**	3779	1342
C	1229	3627	**C**	1265	3651	**C**	1265	3692	**C**	1277	3718	**C**	1221	3658
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3741	1302	**S**	3771	1337	**S**	3744	1331	**S**	3818	1334	**S**	3752	1288
C	1259	3698	**C**	1229	3663	**C**	1256	3669	**C**	1182	3666	**C**	1248	3712
Tiempo total	00:01:06													

Tabla 5.38. SRM, rendimiento de modelos con WOW.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2602	0.7398	0.7435
2	0.2614	0.7386	0.7408
3	0.2573	0.7427	0.7438
4	0.2559	0.7441	0.7442
5	0.2563	0.7437	0.7468
6	0.2561	0.7439	0.7450
7	0.2566	0.7434	0.7461
8	0.2587	0.7413	0.7432
9	0.2516	0.7484	0.7522
10	0.2536	0.7464	0.7474
Media	**0.2568**	**0.7432**	**0.7453**

5.3.3.3.2 Entrenamiento 9 y 10 con payload 0.40 bpp

Algoritmo HILL

En esta prueba se realizó el entrenamiento con el SRM para detectar el algoritmo HILL con un nivel de carga de 0.40 bpp. En la Tabla 5.39 se muestra el número de clasificadores utilizados. En las Tabla 5.40 y 5.41 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.39. SRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	94	90	118	106	94	84	98	111	87	122
Tiempo total (HH:MM:SS)	00:10:59									

Tabla 5.40. SRM, detección de HILL con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3804	1324	**S**	3786	1280	**S**	3803	1278	**S**	3817	1273	**S**	3791	1290
C	1196	3676	**C**	1214	3720	**C**	1197	3722	**C**	1183	3727	**C**	1209	3710
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3740	1299	**S**	3785	1261	**S**	3792	1274	**S**	3819	1279	**S**	3813	1250
C	1260	3701	**C**	1215	3739	**C**	1208	3726	**C**	1181	3721	**C**	1187	3750
Tiempo total	00:00:47													

Tabla 5.41. SRM, rendimiento de modelos con HILL.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.2520	0.7480	0.7512
2	0.2494	0.7506	0.7522
3	0.2475	0.7525	0.7545
4	0.2456	0.7544	0.7566
5	0.2499	0.7501	0.7521

Tabla 5.41. Continuación.

Prueba	P_E	Accuracy	F-Measure
6	0.2559	0.7441	0.7451
7	0.2476	0.7524	0.7535
8	0.2482	0.7518	0.7534
9	0.2460	0.7540	0.7564
10	0.2437	0.7563	0.7578
Media	**0.2486**	**0.7514**	**0.7533**

Algoritmo WOW

En esta prueba se realizó el entrenamiento con el SRM para detectar el algoritmo WOW con 0.40 bpp. En la Tabla 5.42 se muestra el número de clasificadores utilizados. En las Tabla 5.43 y 5.44 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.42. SRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	80	84	88	95	126	89	102	97	84	112
Tiempo total (HH:MM:SS)	00:13:56									

Tabla 5.43. SRM, detección de WOW con 0.40 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4014	1128	**S**	3972	1059	**S**	3942	1004	**S**	4005	1074	**S**	4007	1046
C	986	3872	**C**	1028	3941	**C**	1058	3996	**C**	995	3926	**C**	993	3954
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	3973	1096	**S**	3994	1109	**S**	3995	1049	**S**	4035	1127	**S**	4038	1042
C	1027	3904	**C**	1006	3891	**C**	1005	3951	**C**	965	3873	**C**	962	3958
Tiempo total			00:00:39											

Tabla 5.44. SRM, rendimiento de modelos con WOW.

Prueba	P_E	Accuracy	F-Measure
1	0.2114	0.7886	0.7916
2	0.2087	0.7913	0.7919
3	0.2062	0.7938	0.7927
4	0.2069	0.7931	0.7947
5	0.2039	0.7961	0.7972
6	0.2123	0.7877	0.7892
7	0.2115	0.7885	0.7907
8	0.2054	0.7946	0.7955
9	0.2092	0.7908	0.7941
10	0.2004	0.7996	0.8012
Media	**0.2076**	**0.7924**	**0.7939**

5.3.3.3.3 Entrenamiento 11 y 12 con payload 0.50 bpp

Algoritmo HILL

Esta prueba se efectuó el entrenamiento con el SRM para detectar el algoritmo HILL con un nivel de carga de 0.50 bpp. En la Tabla 5.45 se muestra el número de clasificadores utilizados. En las Tabla 5.46 y 5.57 se exponen las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.45. SRM, número de clasificadores para detectar el algoritmo HILL.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	82	80	84	95	110	96	105	92	100	89
Tiempo total (HH:MM:SS)	00:10:53									

Tabla 5.46. SRM, detección de HILL con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4019	1078	**S**	4038	1040	**S**	4022	1053	**S**	4046	1067	**S**	4043	1034
C	981	3922	**C**	962	3960	**C**	978	3947	**C**	954	3933	**C**	957	3966
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4014	1066	**S**	4059	1090	**S**	4012	1028	**S**	4075	1099	**S**	4033	1022
C	986	3934	**C**	941	3910	**C**	988	3972	**C**	925	3901	**C**	967	3978
Tiempo total			00:00:40											

Tabla 5.47. SRM, rendimiento de modelos con HILL.

Prueba	P_E	Accuracy	F-Measure
1	0.2059	0.7941	0.7961
2	0.2002	0.7998	0.8013
3	0.2031	0.7969	0.7984
4	0.2021	0.7979	0.8002
5	0.1991	0.8009	0.8024
6	0.2052	0.7948	0.7964
7	0.2031	0.7969	0.7999
8	0.2016	0.7984	0.7992
9	0.2024	0.7976	0.8011
10	0.1989	0.8011	0.8022
Media	**0.2022**	**0.7978**	**0.7997**

Algoritmo WOW

En esta prueba se realizó el entrenamiento con el SRM para detectar el algoritmo WOW con 0.50 bpp. En la Tabla 5.48 se muestra el número de clasificadores utilizados. En las Tabla 5.49 y 5.50 se muestran las matrices de confusión con el resultado de las pruebas y los resultados de las métricas, respectivamente.

Tabla 5.48. SRM, número de clasificadores para detectar el algoritmo WOW.

Entrenamiento	1	2	3	4	5	6	7	8	9	10
No. de clasificadores	84	117	84	86	105	116	105	105	84	99
Tiempo total (HH:MM:SS)	00:13:59									

Tabla 5.49. SRM, detección de WOW con 0.50 bpp, S = inyectada, C = limpia.

***Split* 1**			***Split* 2**			***Split* 3**			***Split* 4**			***Split* 5**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4218	986	**S**	4156	880	**S**	4188	833	**S**	4216	919	**S**	4175	900
C	782	4014	**C**	844	4120	**C**	812	4167	**C**	784	4081	**C**	825	4100
***Split* 6**			***Split* 7**			***Split* 8**			***Split* 9**			***Split* 10**		
	S	**C**		**S**	**C**		**S**	**C**		**S**	**C**		**S**	**C**
S	4175	860	**S**	4187	921	**S**	4232	878	**S**	4203	883	**S**	4205	844
C	825	4140	**C**	813	4079	**C**	768	4122	**C**	797	4117	**C**	795	4156
Tiempo total	00:00:47													

Tabla 5.50. SRM, rendimiento de modelos con WOW.

Prueba	**P_E**	**Accuracy**	**F-Measure**
1	0.1768	0.8232	0.8267
2	0.1724	0.8276	0.8282
3	0.1645	0.8355	0.8358
4	0.1703	0.8297	0.8320
5	0.1725	0.8275	0.8288
6	0.1685	0.8315	0.8321
7	0.1734	0.8266	0.8285
8	0.1646	0.8354	0.8372
9	0.1680	0.8320	0.8334
10	0.1639	0.8361	0.8369
Media	**0.1695**	**0.8305**	**0.8320**

5.3.4 Caso de desborde de memoria

Considerando el ambiente de pruebas, específicamente, el tamaño de la memoria RAM y memoria de intercambio. Se recomienda usar, el código que se detalla en el Anexo D. Originalmente, la implementación de los autores para utilizar el clasificador ensamblando, se ejecutaba la parte de entrenamiento y la parte de prueba en conjunto, esto ocasiona que al no tener suficiente capacidad de almacenamiento de memoria RAM, el sistema operativo se detiene o colapsa. Una solución a este problema, fue separar la parte del entrenamiento de la parte de prueba. Para mayor detalle recurrir al Anexo D.

5.4 Análisis de resultados

Para un análisis más cómodo y rápido de los distintos entrenamientos y las correspondientes pruebas realizadas, los resultados se resumen en las Tablas 5.51-5.53, en donde se compara el método propuesto con el enfoque de la literatura con la media de las métricas: error de detección, *Accuracy* y *F-measure*, respectivamente.

Tabla 5.51. Rendimiento de estegoanálisis con métrica P_E.

	$\overline{P}_E$		
	0.30 bpp	0.40 bpp	0.50 bpp
HILL			
SubSRM	0.3217	0.2768	0.2328
Sub2SRM	0.3094	0.2587	0.2172
SRM	0.2991	0.2486	0.2022
WOW			
SubSRM	0.2859	0.2360	0.1954
Sub2SRM	0.2687	0.2218	0.1832
SRM	0.2568	0.2076	0.1695

Tabla 5.52. Rendimiento de estegoanálisis con métrica Accuracy.

	Media Accuracy		
	0.30 bpp	0.40 bpp	0.50 bpp
HILL			
SubSRM	0.6783	0.7232	0.7672
Sub2SRM	0.6906	0.7413	0.7827
SRM	0.7009	0.7514	0.7978
WOW			
SubSRM	0.7141	0.7640	0.8046
Sub2SRM	0.7313	0.7782	0.8168
SRM	0.7432	0.7924	0.8305

Tabla 5.53. Rendimiento de estegoanálisis con métrica F-measure.

	Media F-measure		
	0.30 bpp	0.40 bpp	0.50 bpp
HILL			
SubSRM	0.6437	0.6967	0.7469
Sub2SRM	0.6922	0.7456	0.7860
SRM	0.7025	0.7533	0.7997
WOW			
SubSRM	0.6857	0.7428	0.7896
Sub2SRM	0.7345	0.7809	0.8190
SRM	0.7453	0.7939	0.8320

En la Tabla 5.54 se muestra que los resultados obtenidos con la métrica *error de detección* $\boldsymbol{P_E}$, son casi los mismos que los reportados en la literatura. Los experimentos de los autores citados en la tabla siguieron las mismas condiciones de los conjuntos de entrenamiento y prueba en las imágenes que se reportan en este trabajo de investigación. Los que indica que se replicó el enfoque sin problema.

Tabla 5.54. Replicación del SRM de este trabajo y el reportado en literatura.

	$\overline{P}_E$		
	0.30 bpp	0.40 bpp	0.50 bpp
HILL			
SRM [Zhou, 2018]	0.3002	0.2492	0.2051
SRM (ésta investigación)	0.2991	0.2486	0.2022
WOW			
SRM [Denemark, 2014]	0.2553	0.2060	0.1683
SRM (ésta investigación)	0.2568	0.2076	0.1695

En las siguientes tablas se muestra el tiempo para calcular las características de los dos enfoques de estegoanálisis y también el tiempo al aplicar los algoritmos de esteganografía para generar los conjuntos de imágenes inyectadas, esto se muestra en la Tabla 5.55 y 5.56, respectivamente.

Tabla 5.55. Tiempo de ejecución estegoanálisis.

Entrenamiento	Característica	Tiempo (HH:MM:SS)
1 y 2	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
3 y 4	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
5 y 6	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Cover = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
	Stego = Sub-imagen + SRM	05:58:31 + 06:46:00 = 12:44:31
Total		**152:18:12**
7 y 8	Cover = SRM	11:48:57
	Stego = SRM	11:48:57
	Stego = SRM	11:48:57
9 y 10	Stego = SRM	11:48:57
	Stego = SRM	11:48:57
11 y 12	Stego = SRM	11:48:57
	Stego = SRM	11:48:57
Total		**82:42:39**
Total General		**235:00:51**

Tabla 5.56. Tiempo de ejecución esteganografía.

No.	Algoritmo	Tiempo
1	HILL (10000 imágenes)	00:31:40
2	WOW (10000 imágenes)	00:35:24

Con el redimiendo conseguido de ambos enfoques se concluye lo siguiente.

a) Al comparar el modelo generado con el método propuesto SubSRM y el modelo creado con el SRM, éste último tiene un rendimiento mejor en todos los casos en ambos algoritmos de esteganografía con las tres métricas de evaluación. Los resultados con el enfoque propuesto no son iguales al modelo de comparación pero son cercanos.

b) La mayor diferencia se presentó cuando la carga fue de 0.50 con el algoritmo HILL, la diferencia fue de 0.0306 respecto al *error de detección promedio* P_E en SubSRM. En Sub2SRM la mayor diferencia fue de 0.015. En el algoritmo WOW la mayor diferencia fue de 0.0291 en SubSRM con carga de 0.30 bpp y la mayor fue de 0.0142 en Sub2SRM con carga de 0.40 bpp.

c) Para la métrica promedio accuracy se tuvo un comportamiento similar al punto anterior, las mayores diferencias se presentaron con 0.40 bpp para el algoritmo HILL. Para SubSRM fue de 0.0306 y para Sub2SRM fue 0.0151. Con el algoritmo WOW las mayores diferencias fueron de 0.0291 en SubSRM con 0.30 bpp y en Sub2SRM fue de 0.0142 con 0.40. bpp

d) En la métrica promedio *F-measure* la mayor diferencia con HILL en SubSRM fue de 0.0588 con la carga 0.30 bpp y en Sub2SRM fue de 0.0137 con carga 0.50 bpp. Con el algoritmo WOW las mayores diferencias son cuando la carga es de 0.30 bpp en SubSRM con 0.0596 y se presentó una diferencia de 0.013 en Sub2SRM con las cargas de 0.40 y 0.50 bpp.

e) En la mayoría de los resultados mostrados en la literatura se entrena con 5000 imágenes y se prueba con 5000, esto bajo diez divisiones aleatorias (*Splits*). El nivel del error $\bar{P}_E$ es casi el mismo en comparación con lo reportado, esto se observa en la Tabla 5.54, indicando que la replicación del método fue confiable.

f) El método presentado SubSRM tiene una tendencia para la detección de la clase de imágenes limpias, como se observó en las matrices de confusión de la sección 5.3.3.2, teniendo mejores resultados para todos los casos con ambos algoritmos de esteganografía, en comparación con el SRM. Para la clase inyectada presenta un rendimiento menor respeto al enfoque de la literatura. Esto es más notable para la métrica F-measure, debido a que esta métrica no considera los TN (*True Negative*) o verdaderos negativos.

g) Para la variante Sub2SRM tiene un rendimiento cercano al SRM, para la mayoría de los casos. Esta variante que se prueba con las imágenes que fueron generadas con el método de las sub-imágenes, no tiene una tendencia a detectar la clase limpia sino un poco más a detectar la clase inyectada y en algunos casos, como se observa en las matrices de confusión, tiene un comportamiento equilibrado al equivocarse en las clases.

h) El tiempo para la extracción de características para el conjunto de imágenes limpias e inyectadas para el entrenamiento 1, *cover* y *stego* en la Tabla 5.55, fue de 06:46:00 y 6:46:00 en comparación al entrenamiento 7 que fue de 11:48:57 para las características *cover* y el mismo 11:48:57 para las características *stego*. Esto se logró por el tamaño de la imagen que fue 384x384, para calcular las sub-imágenes de éste tamaño el tiempo fue de 05:58:31 para cada clase de las características.

i) Se observó que al aumentar el tamaño de la sub-imagen la clasificación mejoró pero no se evaluó con dimensiones mayores a 384x384, debido a que el tiempo total de generar las sub-imágenes y extraer características

incrementó respecto al tiempo tomado con el tamaño original del conjunto de imágenes. Por tal motivo, no se asegura que el umbral de 384x384 sea un óptimo global para el método que se presentó.

j) Tal como está reportado en el estado del arte el algoritmo HILL es más resistente al estegoanálisis que el algoritmo WOW, como se puede observar en las Tablas 5.51-5.53 y la Tabla 5.57.

k) Para el caso de Sub2SRM es una forma rápida para medir el comportamiento de clasificación cuando se recibe un objeto no visto, al que se le aplica un mecanismo para obtener la mejor partición de 384x384.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones generales

En esta sección se presentan los objetivos de la investigación con el fin de evaluar el cumplimiento de los mismos en esta tesis. Asimismo, se explican las aportaciones de esta investigación, las limitaciones del enfoque que fue presentado y las posibilidades de realizar instigaciones a futuro.

El objetivo principal de la propuesta de tesis fue:

“Estudiar técnicas de IA aplicables al estegoanálisis de imágenes digitales, y proponer un método o algoritmo desde un enfoque de IA para detectar imágenes inyectadas.”

El objetivo se cumplió, ya que se propuso e implementó un método para obtener una sub-imagen que sumándose con técnicas de procesamiento digital de imágenes y aprendizaje automático se genera un modelo de aprendizaje, que permite la clasificación de un objeto (imagen) no visto. En la Tabla 6.1 se analizan los objetivos específicos planteados al inicio de la investigación y el alcance real.

Tabla 6.1. Objetivos específicos propuestos y su alcance real.

Objetivo específico propuesto	Alcance real
1a. Identificar algoritmos de esteganografía para inyección en imágenes digitales.	Se identificaron varios algoritmos y se seleccionaron dos, HILL y WOW.
2a. Identificar algoritmos de estegoanálisis para la contramedida.	Se identificaron los algoritmos para extraer características los más frecuentes: SPAM, SRM, variantes del SRM y como algoritmo de clasificación: Clasificador ensamblado para estegoanálisis.
3a. Escoger técnicas de Inteligencia Artificial, reportadas en el estado del arte, aplicables al estegoanálisis de imágenes.	Las técnicas seleccionadas fueron el SRM y el clasificador ensamblado para estegoanálisis.
4a. Implementar las técnicas de Inteligencia Artificial seleccionadas.	La implementación de técnicas se hizo en el sistema. Algunas fueron adaptadas para funcionar más eficiente con código el disponible por los autores como el SRM.
5a. Determinar el banco de imágenes para la experimentación y pruebas.	El banco de imágenes que se utilizó fue BOSSbase versión 1.01.
6a. Buscar posibles patrones característicos de las imágenes inyectadas con técnicas de Visión Artificial.	Se buscó maximizar los pixeles modificados para aumentar el rendimiento de patrones para una mayor separabilidad de las clases mediante el método propuesto.

La Tabla 6.2 muestra un análisis de los alcances y limitaciones planeados al inicio de la investigación y el cumplimento real.

Tabla 6.2. Alcances y limitaciones y su alcance real.

Alcances y limitaciones	Alcance real
1b. Usar o adaptar técnicas de esteganografía para crear bancos de imágenes.	Se adaptaron técnicas para optimizar el rendimiento del hardware mediante Subprocesos en el lenguaje Python3.
2b. Evaluar los algoritmos y técnicas de IA y sus enfoques para la detección de esteganografía.	Con lo reportado en la literatura se evaluó el rendimiento con las métricas *Error de detección* P_E , *Accuracy* y *F-measure*.
3b. Escoger al menos tres técnicas de Inteligencia Artificial, aplicables al estegoanálisis de imágenes.	Las técnicas seleccionadas fueron: Método propuesto SubSRM, SRM el Clasificador Ensamblado para Estegoanálisis.
4b. Proponer e implementar un método para identificar imágenes inyectadas.	Se propuso un método para este propósito descrito en la Sección 3.2.
5b. El método solo funcionará con imágenes digitales.	El método solo funciona para imágenes digitales.
6b. Detectar imágenes inyectadas con dos de las técnicas más frecuentes de esteganografía.	Los algoritmos a detectar fueron, HILL y WOW.
7b. Las imágenes serán inyectadas con texto.	Se logró inyectar imágenes con un mensaje de texto personalizado y una clave recibidas por parte del usuario solo para el algoritmo HILL.
8b. Dado que se trabajó con el banco de imágenes BOSSBase 1.01 (Sección 2.4), las características de las imágenes son las siguientes. Escala de grises, y formato pgm.	Se utilizó el banco de imágenes BOSSBase.

En la Tabla 6.3 se validan los objetivos y alcances de este trabajo de investigación referentes a las Tablas 6.1 y 6.2

6.2 Aportación

A continuación se presentan las aportaciones realizadas de esta investigación.

- La aportación principal fue el método propuesto que si bien no ofrece resultados iguales con el modelo comparado, los resultados fueron cercanos y da indicios que ese camino no debe de ser retomado si se busca mejorar el nivel de detección.

- Para el problema planteado no existen trabajos previos para la esteganografía y el estegoanálisis en el Tecnológico Nacional de México / CENIDET. Sin embargo, se decidió realizar la investigación para contrarrestar métodos que están reportados en el estado del arte que son adaptativos y difíciles de detectar, y no con métodos clásicos como lo son el LSB y sus variantes.
- Se presenta un sistema el cual tiene el método propuesto y funciones para una rápida experimentación con algoritmos de esteganografía y extracción de características de estegoanálisis.
- En el sistema el algoritmo de esteganogrfía HILL tiene la funcionalidad para inyectar un mensaje de texto proporcionado por el usuario.
- Para el clasificador ensamblado se dividió el código en entrenamiento y prueba, como se muestra en el Anexo D.

Tabla 6.3. Validación de objetivos y alcances de la tesis referentes a las Tablas 6.1 y 6.2.

Prueba	Alcances										
	Filas de Tabla 6.1						Filas de Tabla 6.2				
	1a	2a	3a	4a	5a	6a	1b	2b	3b	4b	6b
5.3.2 Esteganografía	X				X		X				X
5.3.3.1 Obtención de umbrales		X	X	X		X			X	X	
5.3.3.2 Detección con el algoritmo propuesto			X	X		X		X	X	X	
5.3.3.2.1 Entrenamiento 1 y 2 con payload 0.30 bpp		X	X	X		X			X	X	
5.3.3.2.2 Entrenamiento 3 y 4 con payload 0.40 bpp		X	X	X		X			X	X	
5.3.3.2.3 Entrenamiento 5 y 6 con payload 0.50 bpp		X	X	X		X			X	X	
5.3.3.3.1 Entrenamiento 7 y 8 con payload 0.30 bpp	X	X		X					X		
5.3.3.3.2 Entrenamiento 9 y 10 con payload 0.40 bpp	X	X		X					X		
5.3.3.3.3 Entrenamiento 11 y 12 con payload 0.50 bpp	X	X		X					X		

6.3 Limitaciones del algoritmo propuesto

El algoritmo propuesto tienes algunas limitaciones. La más notable es que al realizar la clasificación detecta mejor la clase limpia que la clase inyectada, lo cual es lo más esperado es que se detecte de forma más precisa la clase *stego*, como sucede con el SRM. En general es mejor que el clasificador se equivoque al detectar una imagen que está limpia como inyectada, a que falle al detectar una imagen que esta inyectada como limpia.

La otra limitación es que podría no generar la mejor partición o la sub-imagen ideal de la imagen procesada. Esto podría ocurrir cuando el algoritmo de esteganografía no sea muy adaptable a ciertas zonas, es decir, que en lugar de obtener una partición de la imagen idónea, obtenga una con una cantidad de inyección de mensaje pobre. Esto debilitaría al no notar muchos cambios entre las sub-imágenes; limpia e inyectada, al realizar en entrenamiento.

6.4 Trabajo futuro

Como trabajos futuros se enlista lo siguiente.

- Para el método presentado, diseñar un mecanismo con filtros paso alto, para obtener una sub-imagen, es decir, hacer el equivalente del mapa de inyección con un filtrado, para no requerir la imagen limpia e inyectada.
- Calcular los parámetros óptimos para el método presentado.
- Para reducir el tiempo en el método al implementar un algoritmo de optimización, específicamente en el apartado (c) de la sección 3.2.2.2.
- Al calcular la sub-imagen en un objeto desconocido (no visto), se podría calcular un mapa de inyección aproximado resaltado las altas frecuencias en la imagen a clasificar.
- Utilizar las probabilidades de modificación del pixel y la función de costo del algoritmo de esteganografía, para mejorar el nivel de la clasificación.

- Buscar nuevos descriptores o modificar los descriptores para estegoanálisis, pero considerando que la alta dimensión de características es importante al momento de separar las clases, esto debido a las modificaciones que realizan los algoritmos adaptables.
- Investigar y utilizar redes neuronales convolucionales que son aplicadas al estegoanálisis, la arquitectura que utilizan y los elementos de que la integran.

Referencias

[Afrakhteh, 2010] M. Afrakhteh, S. Ibrahim, Adaptive steganography scheme using more surrounding pixels, in: Computer Design and Applications, ICCDA, 2010 International Conference on, IEEE, 2010 pp. V1-225-V221-229.

[Anderson, 1998] Anderson, R. J., & Petitcolas, F. A. (1998). On the limits of steganography. IEEE Journal on selected areas in communications, 16(4), 474-481.

[Bas, 2011] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing BOSS. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, Information Hiding, 13th International Conference, volume 6958 of Lecture Notes in Computer Science, pages 59–70, Prague, Czech Republic, May 18–20, 2011.

[Bhattacharyya, 2011] Bhattacharyya, S. (2011). A survey of steganography and steganalysis technique in image, text, audio and video as cover carrier. Journal of global research in computer science, 2(4).

[Bierbrauer, 1998] Bierbrauer, J. (1998). On Crandall's problem. Personal communication available from http://www. ws. binghamton. edu/fridrich/covcodes. pdf.

[Bierbrauer, 2008] Bierbrauer, J., & Fridrich, J. (2008). Constructing good covering codes for applications in steganography. In Transactions on data hiding and multimedia security III (pp. 1-22). Springer, Berlin, Heidelberg.

[Bovik, 2009] Bovik, A. C. (2009). Basic Gray Level Image Processing. In The Essential Guide to Image Processing (pp. 43-68). Academic Press.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

[Chan, 2004] Chan, C. K., & Cheng, L. M. (2004). Hiding data in images by simple LSB substitution. Pattern recognition, 37(3), 469-474. Chan, C. K., & Cheng, L. M. (2004). Hiding data in images by simple LSB substitution. Pattern recognition, 37(3), 469-474.

[Chen, 2013] Chen, L., Shi, Y. Q., Sutthiwan, P., & Niu, X. (2013, October). Non-uniform quantization in breaking HUGO. In International Workshop on Digital Watermarking (pp. 48-62). Springer, Berlin, Heidelberg.

[Crandall, 1998] Crandall, R. (1998). Some notes on steganography. Posted on steganography mailing list, 1-6.

[Denemark, 2014] Denemark, T., Sedighi, V., Holub, V., Cogranne, R., & Fridrich, J. (2014, December). Selection-channel-aware rich model for steganalysis of digital images. In 2014 IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 48-53). IEEE

[Denemark, 2016] Denemark, T., Fridrich, J., & Comesaña-Alfaro, P. (2016). Improving selection-channel-aware steganalysis features. Electronic Imaging, 2016(8), 1-8.

[Filler & Fridrich 2010] Filler, T., & Fridrich, J. (2010). Gibbs construction in steganography. IEEE Transactions on Information Forensics and Security, 5(4), 705-720.

[Filler & Pevny, 2010] T. Filler, T. Pevny, and P. Bas. BOSS. http://boss.gipsa-lab.grenoble-inp.fr/BOSSRank/, July 2010.

[Filler, 2011] Filler, T., Judas, J., & Fridrich, J. (2011). Minimizing additive distortion in steganography using syndrome-trellis codes. IEEE Transactions on Information Forensics and Security, 6(3), 920-935.

[Filler, 2010] Filler, T., Judas, J., & Fridrich, J. (2010, January). Minimizing embedding impact in steganography using trellis-coded quantization. In Media forensics and security II (Vol. 7541, p. 754105). International Society for Optics and Photonics.

[Fridrich ,2006] Fridrich, J., & Soukal, D. (2006). Matrix embedding for large payloads. IEEE Transactions on Information Forensics and Security, 1(3), 390-395.

[Fridrich, 2001] Fridrich J, Goljan M, Du R. Reliable detection of LSB steganography in color and grayscale images. In: Proceedings of the 2001 workshop on multimedia and security new challenges - (MM&Sec '01); 2001. p. 27.

[Fridrich, 2011] Fridrich, J., Kodovský, J., Holub, V., & Goljan, M. (2011, May). Steganalysis of content-adaptive steganography in spatial domain.

In International Workshop on Information Hiding (pp. 102-117). Springer, Berlin, Heidelberg.

[Fridrich, 2012] Fridrich, J., & Kodovsky, J. (2012). Rich models for steganalysis of digital images. IEEE Transactions on Information Forensics and Security, 7(3), 868-882.

[Gonzalez, 1977] Gonzalez, R. C., & Wintz, P. (1977). Digital image processing (Book). Reading, Mass., Addison-Wesley Publishing Co., Inc.(Applied Mathematics and Computation, (13), 451.

[Gutub, 2010] A.A.A. Gutub, Pixel indicator technique for RGB image steganography, J. Emerg. Technol. Web Intell. 2 (2010) 56–64.

[Haralick, 1973] Haralick, R. M., Shanmugam, K., & Dinstein, I. H. (1973). Textural features for image classification. IEEE Transactions on systems, man, and cybernetics, (6), 610-621.

[Holub, 2012] Holub, V., & Fridrich, J. (2012, December). Designing steganographic distortion using directional filters. In 2012 IEEE International workshop on information forensics and security (WIFS) (pp. 234-239). IEEE.

[Holub, 2013] Holub, V., & Fridrich, J. (2013). Random projections of residuals for digital image steganalysis. IEEE Transactions on information forensics and security, 8(12), 1996-2006.

[Holub, 2014] V. Holub, J. Fridrich, and T. Denemark. Universal distortion design for steganography in an arbitrary domain. EURASIP Journal on Information Security, Special Issue on Revised Selected Papers of the 1st ACM IH and MMS Workshop, 2014:1, 2014

[Hong, 2012] W. Hong, T.-S. Chen, C.-W. Luo, Data embedding using pixel value differencing and diamond encoding with multiple-base notational system, J. Syst. Softw. 85 (2012) 1166–1175.

[Johnson, 1998] Johnson, N. F., & Jajodia, S. (1998, September). Steganalysis: The investigation of hidden information. In Information Technology Conference, 1998. IEEE (pp. 113-116). IEEE

[Joseph, 2011] Joseph, A., & Sundaram, V. (2011). Cryptography and steganography–A survey.

[Ker, 2006] Ker, A. D. (2006, July). Batch steganography and pooled steganalysis. In International Workshop on Information Hiding (pp. 265-281). Springer, Berlin, Heidelberg.

[Ker, 2013] Ker, A. D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., & Pevný, T. (2013, June). Moving steganography and steganalysis from the laboratory into the real world. In Proceedings of the first ACM workshop on Information hiding and multimedia security (pp. 45-58).

[Kodovsky, 2014] Kodovský, J., Sedighi, V., & Fridrich, J. (2014, February). Study of cover source mismatch in steganalysis and ways to mitigate its impact. In Media Watermarking, Security, and Forensics 2014 (Vol. 9028, p. 90280J). International Society for Optics and Photonics.

[Kodovsky, 2010] Kodovsky, J., Pevný, T., & Fridrich, J. (2010, January). Modern steganalysis can detect YASS. In Media Forensics and Security II (Vol. 7541, p. 754102). International Society for Optics and Photonics.

[Kodovsky, 2011] Kodovsky, J., Fridrich, J., & Holub, V. (2011). Ensemble classifiers for steganalysis of digital media. IEEE Transactions on Information Forensics and Security, 7(2), 432-444.

[Li, 2013] Li, M., Kulhandjian, M. K., Pados, D. A., Batalama, S. N., & Medley, M. J. (2013). Extracting spread-spectrum hidden data from digital media. IEEE transactions on information forensics and security, 8(7), 1201-1210.

[Li, 2014] Li, B., Wang, M., Huang, J., & Li, X. (2014, October). A new cost function for spatial image steganography. In 2014 IEEE International Conference on Image Processing (ICIP) (pp. 4206-4210). IEEE.

[Li, 2015] Li, B., Wang, M., Li, X., Tan, S., & Huang, J. (2015). A strategy of clustering modification directions in spatial image steganography. IEEE Transactions on Information Forensics and Security, 10(9), 1905-1917.

[Long, 2000] Fridrich, J., & Long, M. (2000, July). Steganalysis of LSB encoding in color images. In 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532) (Vol. 3, pp. 1279-1282). IEEE.

[Luo, 2010] Luo, W., Huang, F., & Huang, J. (2010). Edge adaptive image steganography based on LSB matching revisited. IEEE Transactions on information forensics and security, 5(2), 201-214.

[MacKay, 2003] MacKay, D. J., & Mac Kay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press.

[Manoj, 2010] Manoj, I. V. S. (2010). Cryptography and steganography. International Journal of Computer Applications, 1(12), 63-68.

[Mishra, 2015] Mishra, R., & Bhanodiya, P. (2015, March). A review on steganography and cryptography. In 2015 International Conference on Advances in Computer Engineering and Applications (pp. 119-122). IEEE.

[Mohammadi, 2014] Mohammadi, F. G., & Abadeh, M. S. (2014). Image steganalysis using a bee colony based feature selection algorithm. Engineering Applications of Artificial Intelligence, 31, 35-43.

[Mohammadi, 2017] Mohammadi, F. G., & Sajedi, H. (2017). Region based image steganalysis using artificial bee colony. Journal of Visual Communication and Image Representation, 44, 214-226.

[Pevny & Filler, 2010] Pevny, T., Filler, T., & Bas, P. (2010, June). Using high-dimensional image models to perform highly undetectable steganography. In International Workshop on Information Hiding (pp. 161-177). Springer, Berlin, Heidelberg.

[Pevny, 2007] Pevny, T., & Fridrich, J. (2007, March). Merging Markov and DCT features for multi-class JPEG steganalysis. In Security, Steganography, and Watermarking of Multimedia Contents IX (Vol. 6505, p. 650503). International Society for Optics and Photonics.

[Pevny, 2010] T. Pevny, P. Bas, J. Fridrich, Steganalysis by subtractive pixel adjacency matrix, IEEE Trans. Inf. Forensics Secur. 5 (2010) 215–224.

[Potdar, 2004] V.M. Potdar, E. Chang, Grey level modification steganography for secret communication, in: Industrial Informatics, 2004 INDIN'04 2004 2nd IEEE International Conference on, IEEE, 2004, pp. 223–228.

[Pradhan, 2016] A. Pradhan, A. K. Sahu, G. Swain and K. R. Sekhar, "Performance evaluation parameters of image steganography techniques," 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), Bangalore, 2016, pp. 1-8.

[Sajedi, 2016] Sajedi, H. (2016). Steganalysis based on steganography pattern discovery. Journal of information security and applications, 30, 3-14.

[Schönfeld, 2006] Schönfeld, D., & Winkler, A. (2006, September). Embedding with syndrome coding based on BCH codes. In Proceedings of the 8th workshop on Multimedia and security (pp. 214-223).

[Sedighi, 2015] Sedighi, V., Cogranne, R., & Fridrich, J. (2015). Content-adaptive steganography by minimizing statistical detectability. IEEE Transactions on Information Forensics and Security, 11(2), 221-234.

[Simmons, 1984] Simmons, G. J. (1984). The prisoners' problem and the subliminal channel. In Advances in Cryptology (pp. 51-67). Springer, Boston, MA.

[Tang, 2014] Tang, W., Li, H., Luo, W., & Huang, J. (2014, June). Adaptive steganalysis against WOW embedding algorithm. In Proceedings of the 2nd ACM workshop on Information hiding and multimedia security (pp.91-96). ACM.

[Tang, 2016] Tang, W., Li, H., Luo, W., & Huang, J. (2016). Adaptive steganalysis based on embedding probabilities of pixels. IEEE Transactions on Information Forensics and Security, 11(4), 734-745

[Van, 2001] Van Dijk, M., & Willems, F. (2001, May). Embedding information in grayscale images. In Proceedings of the 22nd Symposium on Information and Communication Theory in the Benelux, Enschede, The Netherlands (pp. 147-154).

[Wang, 2006] R. Z. Wang, Y.-S. Chen, High-payload image steganography using two-way block matching, IEEE Signal Process. Lett. 13 (2006) 161–164.

[Westfeld, 1999] A. Westfeld, A. Pfitzmann, Attacks on steganographic systems, in: International workshop on information hiding, Springer, 1999, pp. 61–76.

[Westfeld, 2001] Steganalysis, H. C. D. B., & Westfeld, A. (2001, November). F5—a steganographic algorithm. In Information Hiding: 4th International Workshop, IH 2001, Pittsburgh, PA, USA, April 25-27, 2001. Proceedings (Vol. 2137, p. 289). Springer.

[Wu, 2003] D.-C. Wu, W.-H. Tsai, A steganographic method for images by pixel value differencing, Pattern Recognit. Lett. 24 (2003) 1613–1626.

[Yang, 2009] H. Yang, X. Sun, G. Sun, A high-capacity image data hiding scheme using adaptive LSB substitution, Radioengineering 18 (2009) 509–516.

[Zhang, 2006] X. Zhang, S. Wang, Efficient steganographic embedding by exploiting modification direction, IEEE Commun. Lett. 10 (2006) 781–783.

[Zhang, 2009] Zhang, W., & Wang, X. (2009). Generalization of the ZZW embedding construction for steganography. IEEE Transactions on Information Forensics and Security, 4(3), 564-569.

[Zhou, 2018] Zhou, S., Tang, W., Tan, S., & Li, B. (2018, October). Content-Adaptive Steganalysis via Augmented Utilization of Selection-Channel Information. In International Workshop on Digital Watermarking (pp. 261-274). Springer, Cham

Anexo A. Ejemplo STC

En este apartado se explica un ejemplo de codificación de un mensaje en un objeto portador. Esta sección es una versión detallada del ejemplo de Filler en [Filler, 2010].

En mensaje a ocultar es el vector **m** = (0, 1, 1, 1), en un objeto portador **x** = (1, 0, 1, 1, 0, 0, 0, 1). Como resultado el objeto inyectado es **y** = (0, 0, 1, 1, 1, 0, 0, 1). Primero se tiene la matriz $\hat{\mathbb{H}}$, con la cual se puede obtener $\mathbb{H}$, en la Figura A.1 se muestran las matrices. En la figura A.2 se ilustran los elementos de esta codificación.

$$\hat{\mathbb{H}} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathbb{H} = \begin{pmatrix} 1 & 0 & & & \\ 1 & 1 & 1 & 0 & \\ & & 1 & 1 & 1 & 0 \\ & & & & 1 & 1 & 1 & 0 \end{pmatrix}$$

Figura A.1. Matriz $\hat{\mathbb{H}}$ y su correspondiente $\mathbb{H}$.

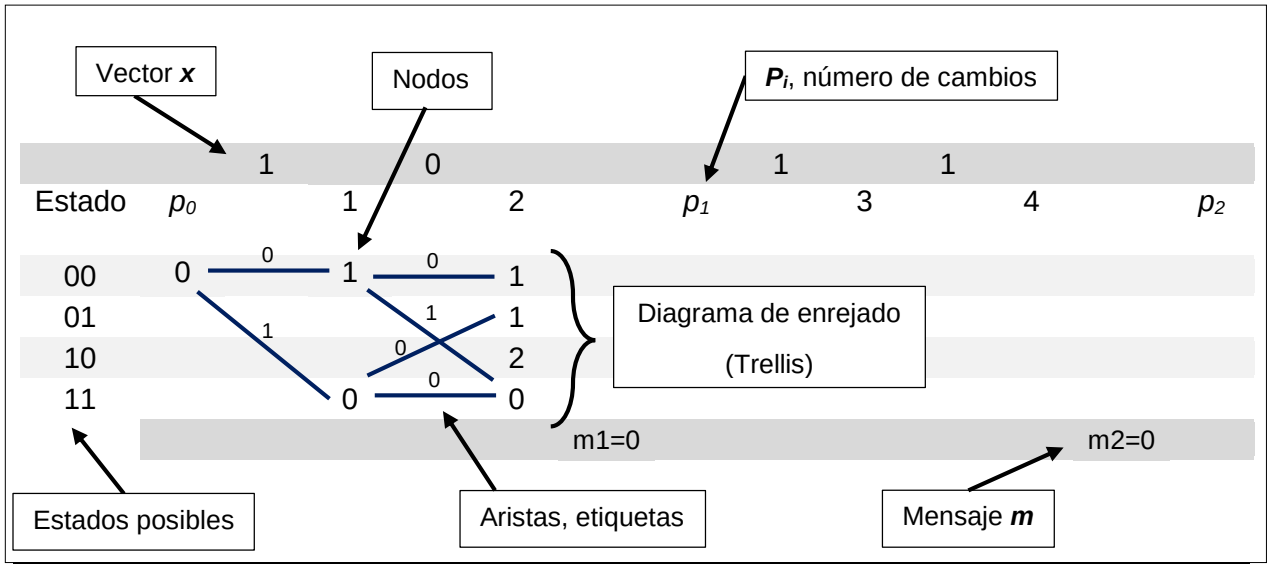


Figura A.2. Elementos de STC.

La codificación se constituye con el algoritmo de Viterbi que funciona con dos partes, hacia delante o *forward* y hacia atrás o *backward*. En la parte *forward*, se encarga de ir creando el enrejado e ir construyendo en camino más corto, una vez que llega a la parte final, el lado derecho del enrejado, se conoce la ruta más corta. En la parte *backward*, se sigue el camino más corto hacia atrás, es decir, de la parte derecha a la izquierda del enrejado y el objeto inyectado más cercano ***y*** es recuperado de las etiquetas de las aristas/bordes.

Forward

Se tiene que minimizar el número de cambios de incrustación, que corresponden a un perfil constante*,* ***pi*** para toda ***i***, es decir, que los cambios tendrán un valor de 1. Considerando la matriz $\mathbb{H}$ mostrada en la Figura A.1, el algoritmo inicia en la columna izquierda, denotada con p_0, el único estado alcanzable que inicia todo en cero, como se ilustra en la Figura A.3.

Dos aristas salen de cada estado (nodo). Para este primer caso, estas dos aristas son el resultado de no agregar y de agregar la primera columna de la matriz $\mathbb{H}$. Cuando se agrega la columna de la matriz $\mathbb{H}$ se aplica la operación XOR entre la columna y el estado. La primer arista conecta el nodo p_0E_{00} (p_0, estado 00) con el nodo C_1E_{00} (columna 1, estado 00). Esta arista es etiquetada con “0” dado que no agrega la primer columna de la matriz $\mathbb{H}$ (ver Figura A.3). Los pesos de los nodos se calculan sumando el peso del nodo inmediato anterior, más el valor del peso del nodo 0 o 1. El peso del nodo va relacionado a la etiqueta, cuando la etiqueta es distinta del valor de x_i, entonces el peso es 1. Como la etiqueta es “0”, significa que $y_1 = 0$, lo que implica que el valor de $x_1 = 1$ cambia a $y_1 = 0$, el bit será diferente. El caso contrario, cuando $y_i == x_i$, el peso es de 0.

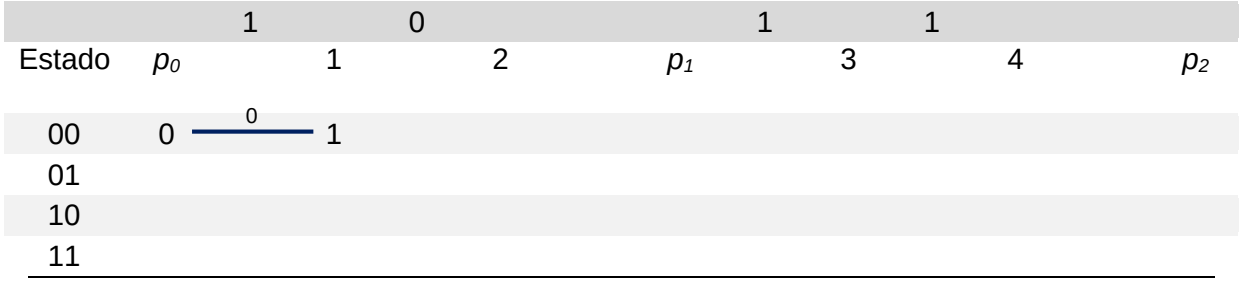


Figura A.3. Forward 1.

La segunda arista inicia en p_0E_{00} (p_0, estado 00) y termina en el nodo C_1E_{11} (columna 1, estado 11). Cambia al estado 11 al agregar la primer columna de $\mathbb{H}$, se le asigna la etiqueta de “1” y el peso es de 0, debido a que $\mathbf{y_1}$== $\mathbf{x_1}$. Esto se observa en la Figura A.4.

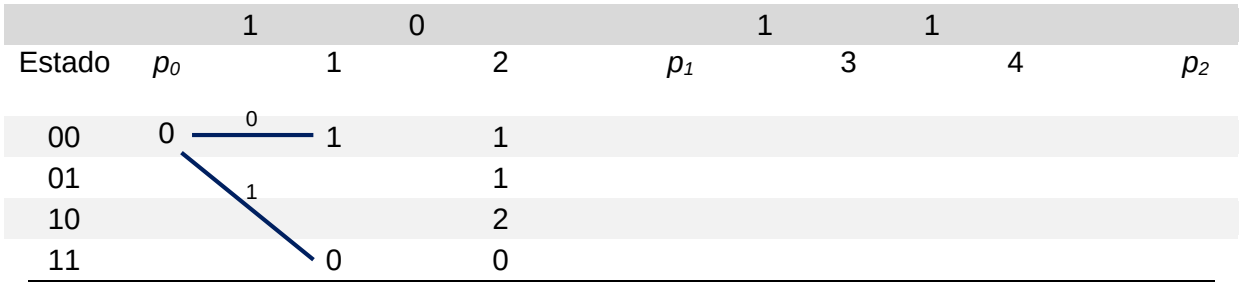


Figura A.4. Forward 2.

Para continuar con las rutas, del nodo C_1E_{00} (columna 1, estado 00) salen dos aristas. La primera conecta con el nodo C_2E_{00} (columna 2, estado 00), al no agregar la segunda columna de $\mathbb{H}$ se asigna la etiqueta “0” y un peso de 0, esto porque $\mathbf{y_2}$== $\mathbf{x_2}$, pero sumando el peso acumulado el peso pasa a 1. La segunda arista va del nodo C_1E_{00} (columna 1, estado 00) al nodo C_2E_{10} (columna 2, estado 10), al agregar la segunda columna de $\mathbb{H}$ se asigna la etiqueta “1” y el peso es 1, dado que $\mathbf{x_2}$ = 0 cambia a $\mathbf{y_2}$ =1, sumado el peso acumulado el peso del nodo es de 2. (Ver Figura A.5).

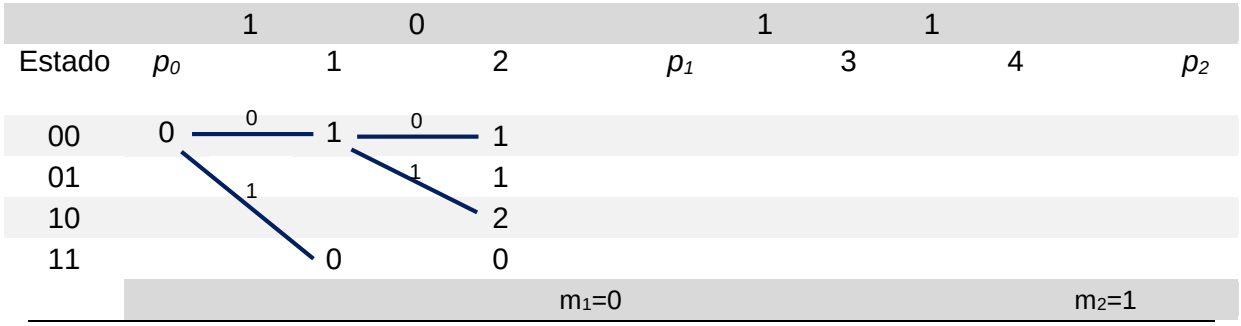


Figura A.5.Forward 3.

Los nodos alcanzables desde el nodo C_1E_{11} (columna 1, estado 11) son dos. La primer arista conecta al nodo C_2E_{00} (columna 2, estado 11), al no agregar la segunda columna de $\mathbb{H}$ se asigna la etiqueta “0” y un peso de 0, esto debido a que

$\mathbf{y_2}$ == $\mathbf{x_2}$, sumando el peso acumulado el peso se queda en 0. La segunda arista va del nodo $\boldsymbol{C_1E_{11}}$ (columna 1, estado 11) al nodo $\boldsymbol{C_2E_{01}}$ (columna 2, estado 01), al agregar la segunda columna de $\mathbb{H}$ se asigna la etiqueta "1" y el peso es 1, dado que $\mathbf{x_2}$ = 0 cambia a $\mathbf{y_2}$ =1, sumado el peso acumulado el peso del nodo es de 1. (Ver Figura A.6).

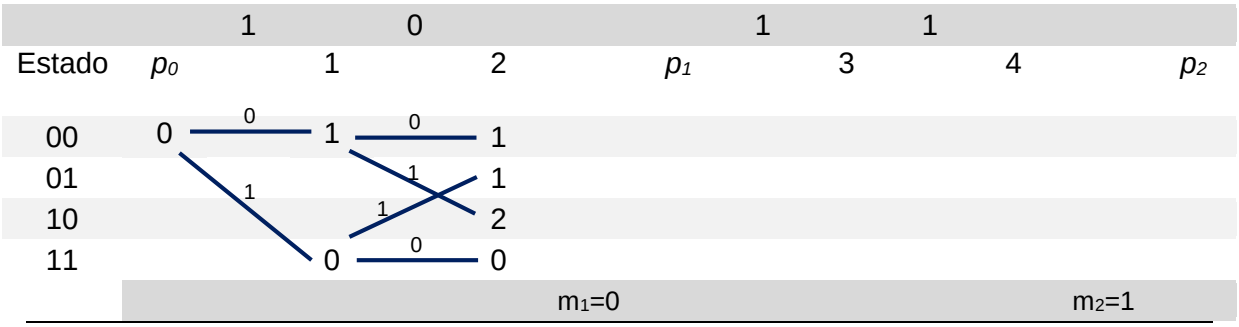


Figura A.6. Forward 4.

Como ya se procesaron las dos primeras columnas de $\mathbb{H}$, el primer bit de ***m*** ya tiene que estar establecido en el valor deseado (en este caso $\mathbf{m_1}$=0). En este punto los últimos nodos tienen los estados siguientes:

- C_2E_{00} *: 00*
- C_2E_{01} *: 0**1** (bit menos significativo**)***
- C_2E_{10} *: 10*
- C_2E_{11} *: 1**1** (bit menos significativo)*

Esto indica que los nodos $\boldsymbol{C_2E_{01}}$ y $\boldsymbol{C_2E_{11}}$ quedan descartados porque el bit menos significativo de ambos es diferente con el bit del mensaje $\mathbf{m_1}$ = 0. Por lo tanto, no salen aristas de estos nodos.

Después, el estado del enrejado pasa de representar los bits (s1, s2) a los bits (s2, s3). Esta transición se puede observar en el Figura A.7, en la columna p_1, y las aristas que la conectan a la columna anterior. Estas aristas no tienen etiqueta y tienen un peso de 0, y representa la poda.

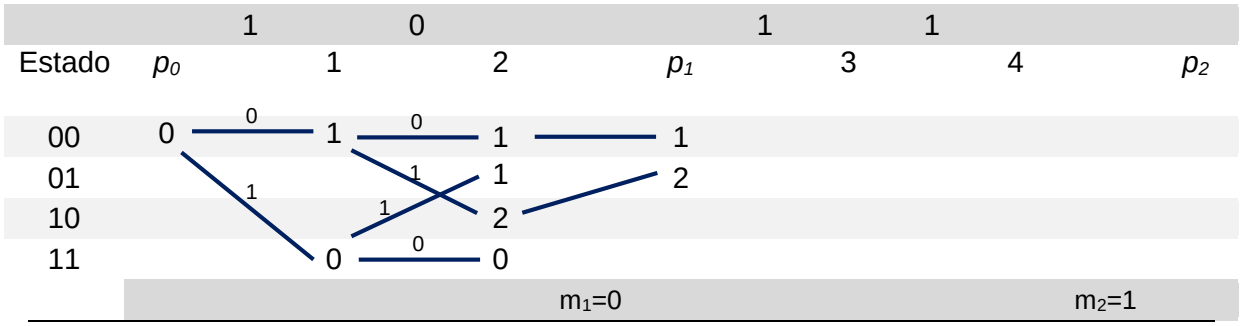


Figura A.7. Forward 5.

La columna 3, se construye de la misma forma que la columna 1 y 2. Se muestra en la Figura A.8.

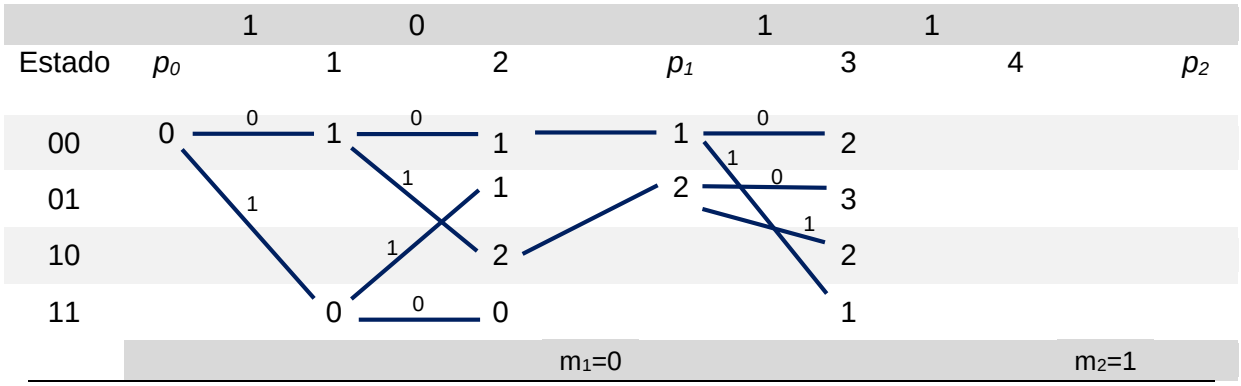


Figura A.8. Forward 6.

La columna 4 se muestra en la Figura A.9, en donde los nodos ***C_3E_{00}*** y ***C_3E_{01}*** salen las aristas que cubren todos los nodos de la columna 4, lo que ocasionará que a un nodo le lleguen dos aristas.

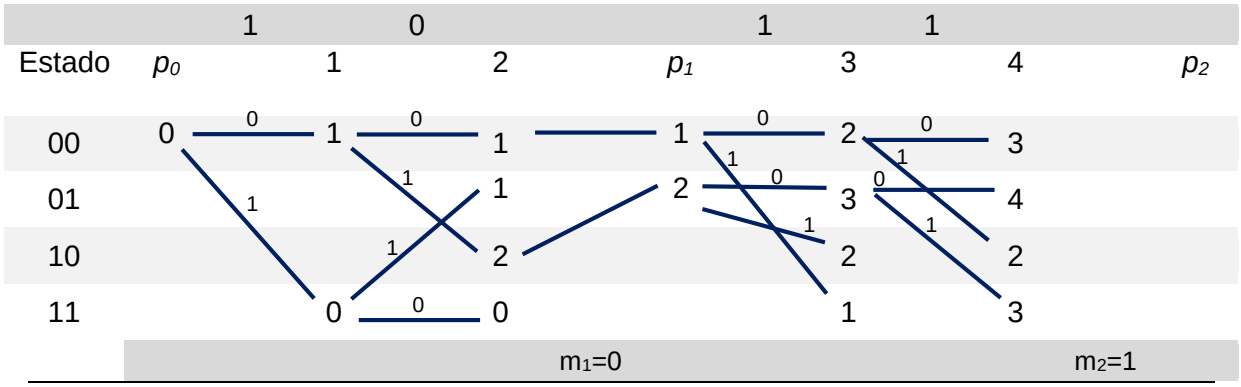


Figura A.9. Forward 7.

En la Figura A.10 se muestra el desarrollo de las aristas que salen de los nodos C_3E_{10} y C_3E_{11}. Mediante los pesos se decide cual es la arista que se queda conectada al nodo siguiente. Este proceso se muestra en la Figura A.11.

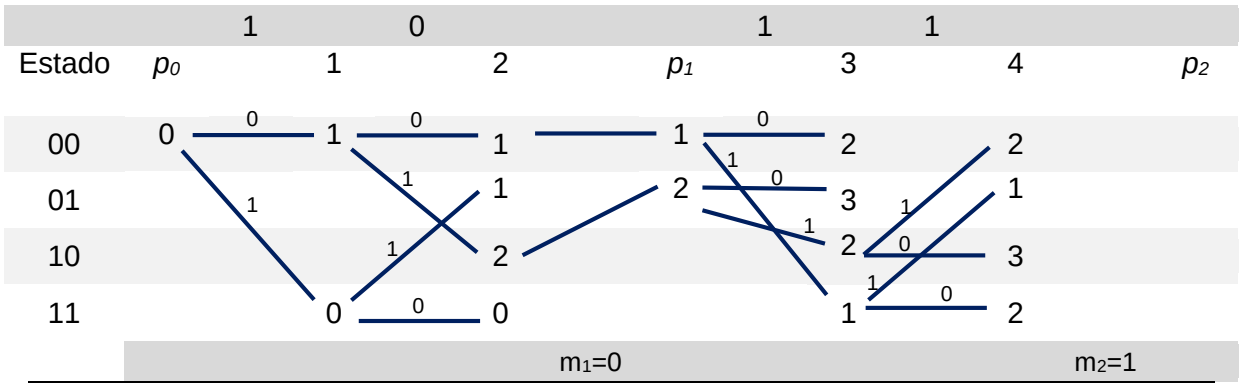


Figura A.10. Forward 8.

El proceso continua de forma análoga, como se muestra en la Figura A.11 y la Figura A.12.

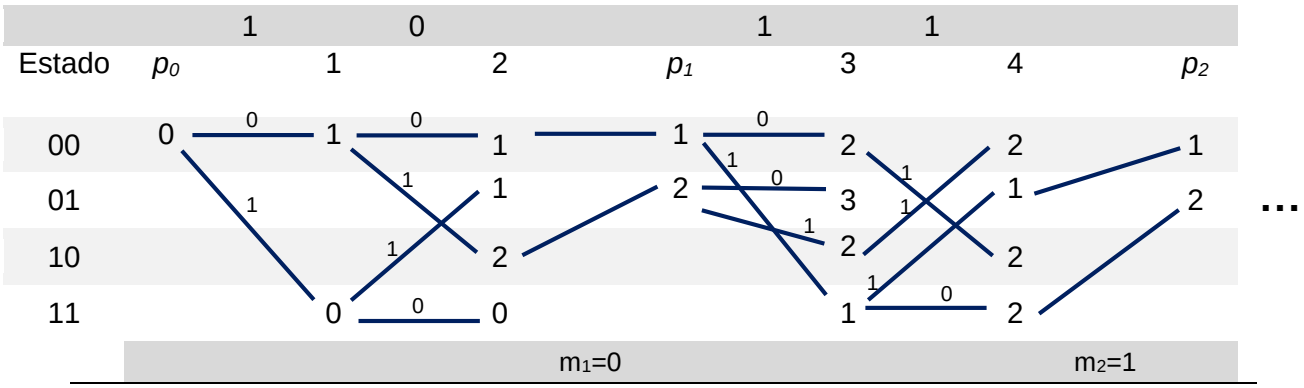


Figura A.11. Forward 9.

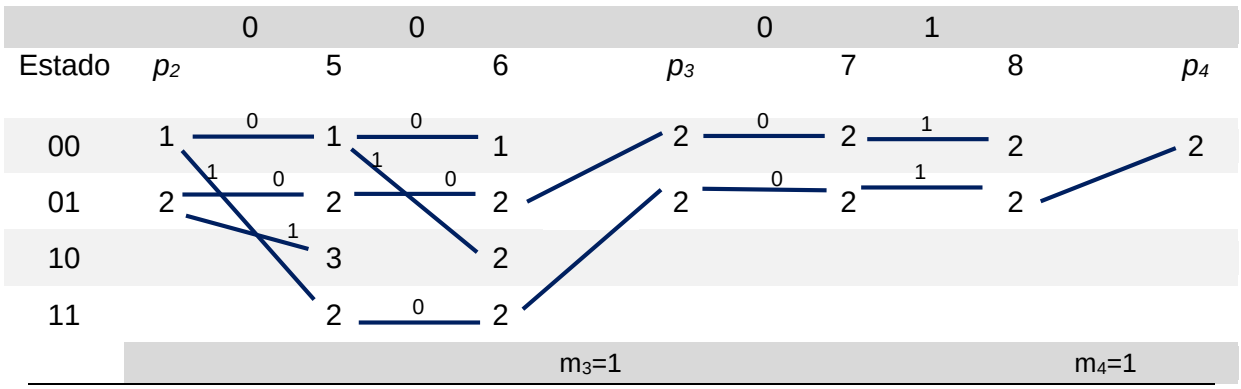


Figura A.12. Forward 10.

En la última parte del enrejado solo se tiene dos estados debido al recorte de la matriz $\mathbb{H}$, lo que asegura que solo quede un estado al terminar el enrejado.

Backward

Cuando la parte Forward se termina, es decir, cuando se llega al final del Trellis. Se comienza a recorrer el enrejado de la parte derecha hacia la parte de inicio por las aristas que no fueron eliminadas y se construye el objeto inyectado ***y*** al leer las etiquetas de las aristas. Es importante mencionar que las aristas de la columna 8 se etiquetan como “1” aunque estén en horizontal, porque corresponde a agregar la columna cero. En las Figuras A.11 y A.12 se muestra el Trellis completo. Al terminar de recorrer la ruta más corta hacia atrás se obtiene el objeto inyectado ***y*** = (0, 0, 1, 1, 1, 0, 0,1).

Anexo B. Modelo Enriquecido Espacial

En esta sección se presenta una explicación de cómo se calcula el descriptor global Modelo Enriquecido Espacial (SRM, *Spatial Rich Model*).

1) Residuos de ruido. Tal como se presentó en el estado del arte en la sección 2.2.2.1, el descriptor comienza con el cálculo del residuo de Ruido $\boldsymbol{R_{ij}}$, para ello se utiliza la ecuación (2.12) de la sección ya mencionada. En esta explicación se utiliza la imagen de la Figura B.1, que tiene una dimensión de 20 filas x 14 columnas (*M* x *N*).

Se realiza el filtrado entre la imagen y el kernel [-1 1], correspondiente a spam14hv, el resultado es mostrado en la Figura B.2.

26	26	26	26	26	26	25	28	51	60	59	57	60	64
25	26	26	26	26	26	25	29	60	73	69	67	69	72
25	26	26	26	26	26	25	27	45	52	50	46	47	49
25	26	25	25	26	26	26	26	31	33	33	31	32	34
24	25	25	25	25	26	26	26	34	39	38	37	37	38
24	25	25	25	25	25	25	26	48	54	51	50	46	53
23	24	25	25	25	26	25	26	47	51	48	46	46	52
23	24	25	25	25	25	25	25	27	26	25	23	25	27
22	23	24	25	25	25	25	25	23	21	22	23	24	25
22	23	24	25	25	25	25	24	37	39	43	50	50	51
20	21	22	22	22	22	21	23	45	53	55	58	58	59
21	22	22	22	22	23	23	23	35	40	39	40	40	40
21	22	22	23	23	23	24	25	24	25	24	24	24	24
32	32	33	34	34	34	29	24	23	23	23	23	22	22
31	31	31	31	30	29	31	24	22	22	22	21	21	21
19	19	19	19	21	21	23	24	21	19	19	19	18	19
21	21	22	22	23	23	23	23	21	18	18	18	18	18
20	20	20	21	22	23	23	23	20	17	16	16	16	16
20	20	20	21	22	22	22	23	19	13	12	12	12	12
20	19	20	21	22	22	22	22	18	11	10	10	10	10

Figura B.1. Imagen de trabajo.

0	0	0	0	-1	4	31	13	-4	-2	2	3
0	0	0	0	-1	2	18	7	-2	-4	1	2
-1	0	1	0	0	0	5	2	0	-2	1	2
0	0	0	1	0	0	8	5	-1	-1	0	1
0	0	0	0	0	1	22	6	-3	-1	-4	7
1	0	0	1	-1	1	21	4	-3	-2	0	6
1	0	0	0	0	0	2	-1	-1	-2	2	2
1	1	0	0	0	0	-2	-2	1	1	1	1
1	1	0	0	0	-1	13	2	4	7	0	1
1	0	0	0	-1	2	22	8	2	3	0	1
0	0	0	1	0	0	12	5	-1	1	0	0
0	1	0	0	1	1	-1	1	-1	0	0	0
1	1	0	0	-5	-5	-1	0	0	0	-1	0
0	0	-1	-1	2	-7	-2	0	0	-1	0	0
0	0	2	0	2	1	-3	-2	0	0	-1	1
1	0	1	0	0	0	-2	-3	0	0	0	0
0	1	1	1	0	0	-3	-3	-1	0	0	0
0	1	1	0	0	1	-4	-6	-1	0	0	0

Figura B.2. Residuo de ruido ***R***.

2) Truncamiento y cuantificación. El siguiente paso es aplicar la ecuación (2.13), para este ejemplo; ***q =1 y T =2***. La operación de truncamiento normaliza los valores de los residuos que están por debajo de ***–T*** y por arriba de ***T***, en el intervalo [-2, 2]. Al aplicar estas operaciones el resultado se manifiesta en la Figura B.3.

0	0	0	0	-1	2	2	2	-2	-2	2	2
0	0	0	0	-1	2	2	2	-2	-2	1	2
-1	0	1	0	0	0	2	2	0	-2	1	2
0	0	0	1	0	0	2	2	-1	-1	0	1
0	0	0	0	0	1	2	2	-2	-1	-2	2
1	0	0	1	-1	1	2	2	-2	-2	0	2
1	0	0	0	0	0	2	-1	-1	-2	2	2
1	1	0	0	0	0	-2	-2	1	1	1	1
1	1	0	0	0	-1	2	2	2	2	0	1
1	0	0	0	-1	2	2	2	2	2	0	1
0	0	0	1	0	0	2	2	-1	1	0	0
0	1	0	0	1	1	-1	1	-1	0	0	0
1	1	0	0	-2	-2	-1	0	0	0	-1	0
0	0	-1	-1	2	-2	-2	0	0	-1	0	0
0	0	2	0	2	1	-2	-2	0	0	-1	1
1	0	1	0	0	0	-2	-2	0	0	0	0
0	1	1	1	0	0	-2	-2	-1	0	0	0
0	1	1	0	0	1	-2	-2	-1	0	0	0

Figura B.3. Truncamiento y cuantificación de ***R***.

3) Co-ocurrencias. Este es el paso importante para generar la característica. Se utiliza $T = 2$, en un rango de $\{-T,\ldots, T\}^4$, es decir, {-2, -1, 0, 1, 2}, que da como resultante el array o arreglo $(2T+1)^4 = 625$ elementos. Se calcula con la ecuación (2.14). A nivel de código se representa por medio de un arreglo de 4 dimensiones, en matlab se representa de la siguiente forma en la Tabla B.1, donde f y c representan la fila y la columna, respectivamente de la matriz interna.

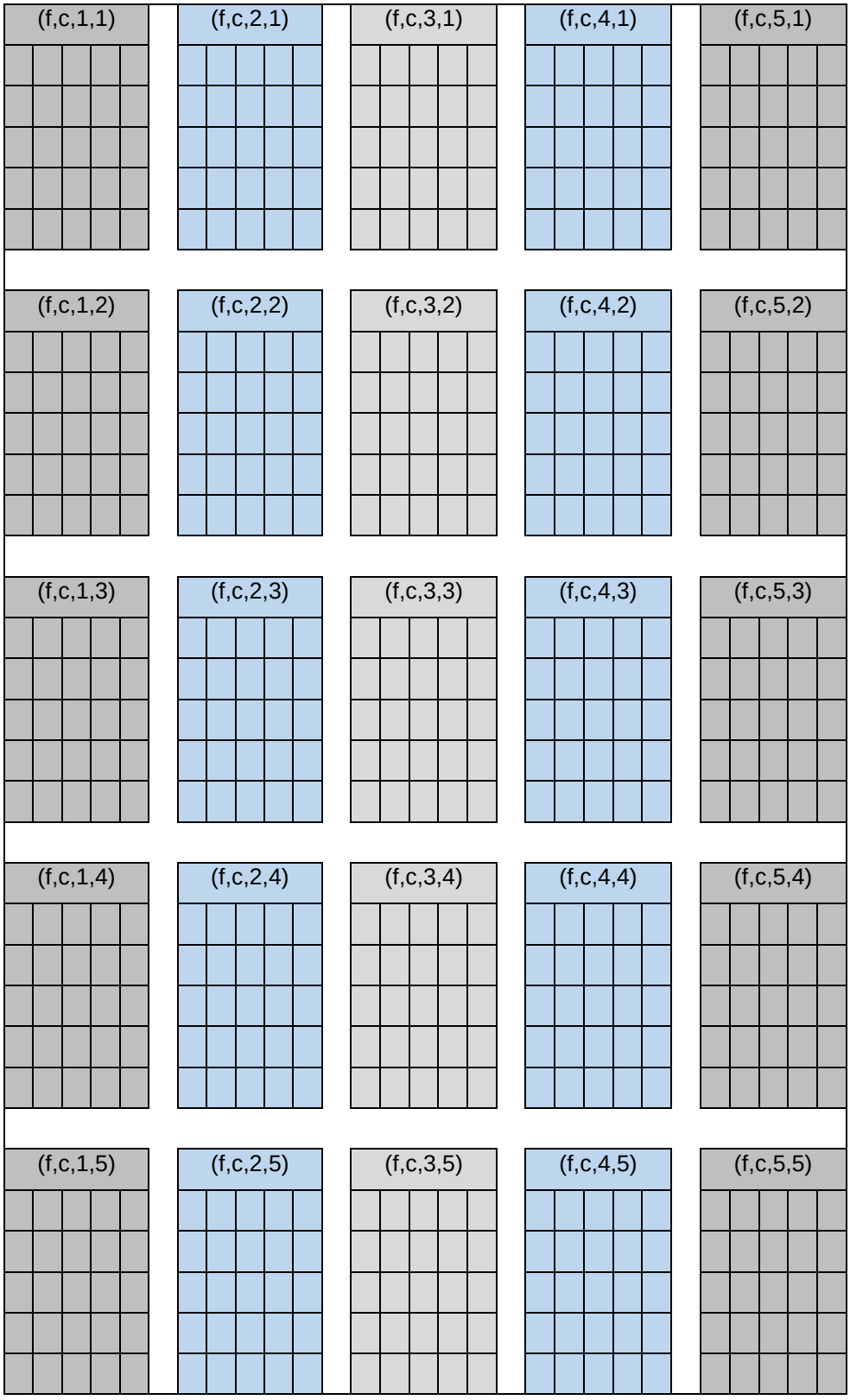
Para calcular las co-ocurrencias nos apoyaremos del seudocódigo de la Figura B.4. Básicamente, son cuatro sentencias *for*, para poder ir manejando el grupo de cuatro residuos de ruido consecutivos, con los índices para formar las co-ocurrencias. Después, el cálculo se guarda en el del array 4-D.

```
1   for i desde -T hasta T  // -2, -1, 0, 1, 2
2     for j desde -T hasta T
3       for k desde -T hasta T
4         for l desde -T hasta T
5           Calcular coocurrencia con el grupo (i,j,k,l)
6           Guardar co-ocurrencias en array4-D con
            index (i+T+1, j+T+1, k+T+1, l+T+1)
7         end // fin for linea 4
8       end // fin for línea 3
9     end // fin for línea 2
10  end // fin for línea 1
```

Figura B.4. Seudocódigo computo de co-ocurrencias.

Por ejemplo, cuando inicia el cálculo, los índices i, j, k, l, tienen el mismo valor de -2, por lo que la primer exploración de la co-ocurrencia es (-2, -2, -2, -2) en el residuo ***R*** de la Figura B.3. Los primeros cinco grupos para explorar, correspondientes al *for l* de la línea 4 (ver Figura B.4) son los siguientes: (-2, -2, -2, -2) (-2, -2, -2, -1), (-2, -2, -2, 0), (-2, -2, -2, 1) y (-2, -2, -2, 2). Desafortunadamente no existen coincidencias en el residuo de este ejemplo, por lo cual no modifican al arreglo.

Tabla B.1. Array 4 dimensiones.



Cuando *k* = -1 en el *for k* del seudocódigo, se obtienen los siguientes grupos en la Tabla B.2. En la figura B.5, se muestran las coincidencias. La matriz que se modifica se ilustra en la Tabla B.3, ésta es parte del arreglo 4-D y se localiza por los últimos dos valores del índice. Este proceso se repite hasta que se calculan los 625 elementos y las matrices del arreglo se llenan (si hay coincidencias), para simplificar el proceso solo se muestran los resultados de la Tabla B.3, en la Tabla B.4, cuándo el proceso ha terminado.

Tabla B.2. Grupos de residuos.

Grupo	Coincidencias	Índice del arreglo
(-2, -2, -1, -2)	0	(1,1,2,1)
(-2, -2, -1, -1)	0	(1,1,2,2)
(-2, -2, -1, 0)	3	(1,1,**2,3**)
(-2, -2, -1, 1)	0	(1,1,2,4)
(-2, -2, -1, 2)	0	(1,1,2,5)

0	0	0	0	-1	2	2	2	-2	-2	2	2
0	0	0	0	-1	2	2	2	-2	-2	1	2
-1	0	1	0	0	0	2	2	0	-2	1	2
0	0	0	1	0	0	2	2	-1	-1	0	1
0	0	0	0	0	1	2	2	-2	-1	-2	2
1	0	0	1	-1	1	2	2	-2	-2	0	2
1	0	0	0	0	0	2	-1	-1	-2	2	2
1	1	0	0	0	0	-2	-2	1	1	1	1
1	1	0	0	0	-1	2	2	2	2	0	1
1	0	0	0	-1	2	2	2	2	2	0	1
0	0	0	1	0	0	2	2	-1	1	0	0
0	1	0	0	1	1	-1	1	-1	0	0	0
1	1	0	0	-2	-2	-1	0	0	0	-1	0
0	0	-1	-1	2	-2	-2	0	0	-1	0	0
0	0	2	0	2	1	-2	-2	0	0	-1	1
1	0	1	0	0	0	-2	-2	0	0	0	0
0	1	1	1	0	0	-2	-2	-1	0	0	0
0	1	1	0	0	1	-2	-2	-1	0	0	0

Figura B.5. Coincidencias encontradas.

Tabla B.3. Matriz de co-ocurrencias inicial.

(f,c,2,3)				
3	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Tabla B.4. Matriz de co-ocurrencias completa.

(f,c,2,3)				
3	0	0	0	0
0	0	0	**1**	0
0	0	**2**	0	0
0	0	0	0	0
0	**1**	0	0	0

Para terminar se suma todos los elementos del arreglo y cada elementos se divide entre el total de la suma, como se observa en la Tabla B.4. Finalmente cada matriz se convierte en un vector y se concatenan, para formar el sub-modelo de tamaño 625. Algunos sub-modelos están formados de más de una co-ocurrencia. Este ejemplo fue para generar la co-ocurrencia horizontal, el procedimiento es análogo para la co-ocurrencia vertical. El modelo enriquecido espacial está formado con los sub-modelos concatenados.

Tabla B.5. Matriz de co-ocurrencias final.

(f,c,2,3)				
0.0185	0	0	0	0
0	0	0	**0.0062**	0
0	0	**0.0123**	0	0
0	0	0	0	0
0	**0.0062**	0	0	0

Anexo C. Clasificador Ensamblado Para Estegoanálisis

El clasificador ensamblado está construido como un *Random forest* que fusiona las decisiones de los clasificadores. Estos clasificadores son entrenados independientemente con un conjunto de imágenes limpias e inyectadas. Cada clasificador es entrenado en un subespacio de características seleccionando de forma aleatoria. Dado un ejemplo del conjunto de pruebas, un objeto a clasificar, la clasificación se genera mediante la votación de todos los clasificadores.

El clasificador depende de dos parámetros d_{sub} y L. El primer parámetro d_{sub} es la dimensión del subespacio del conjunto de características completo d. El segundo parámetro L es la cantidad de clasificadores. Estos dos parámetros se pueden determinar de forma automática como se explica a continuación.

Los dos parámetros afectan el rendimiento del clasificador ensamblado, esto se observa en la Figura C.1. En algoritmo nsF5 (no-shrinkage F5) funciona para imagines JPEG, el payload se fijó a 0.1 bpac (*bits per nonzero AC DCT coefficient*). Las imágenes se dividieron aleatoriamente en dos mitades para entrenamiento y prueba, respectivamente.

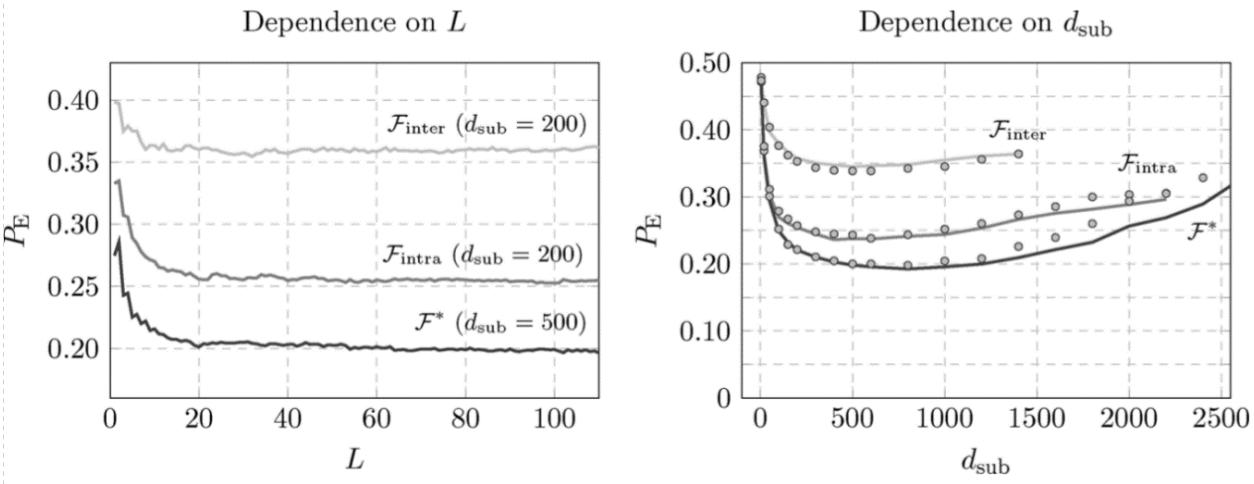


Figura C.1. Comportamiento de L, y d_{sub}. Los puntos el grafica derecha representan el error estimado OOB (out-of-bag). Los conjuntos son, F_{inter}, F_{intra} y F* con dimensiones 1550, 2375, y 3925. El algoritmo de incrustación es nsF5 con payload 0.1 (bpac) [Kodovsky, 2011].

En la gráfica izquierda (ver Fig. C.1) el error P_E se satura rápidamente al aumentar el número de clasificadores L, en el conjunto de prueba. El error P_E (después de la saturación) se muestra en función de la dimensionalidad del subespacio d_{sub} en la Figura C.1 (derecha). Se observa una caída inicial rápida, seguida de un comportamiento constate plano y después el error comienza a crecer nuevamente.

Los autores mencionaron que esto se debe principalmente a las dos razones siguientes. Primero, los clasificadores individuales se vuelven más dependientes y, por lo tanto, la capacidad del clasificador ensamblado para formar límites no lineales disminuye. En segundo lugar, los clasificadores FLD individuales comienzan a sufrir de sobre entrenamiento a medida que aumenta la dimensionalidad del subespacio mientras que el tamaño del entrenamiento sigue siendo el mismo.

Búsqueda de parámetros

Cada clasificador B_l es entrenado con un conjunto (subespacio) {1,…, N^{trn}}, con aproximadamente el 63% de muestras únicas, el 37% restante se utilizan para la validación. La siguiente ecuación es una estimación no sesgada del error de prueba conocido como error fuera de bolsa o “out-of-bag” (OOB):

$$E_{OOB}^{(n)} = \frac{1}{2N^{trn}} \sum_{m=1}^{N^{trn}} \left(B^{(n)}\left(\boldsymbol{x}_m\right) + 1 - B^{(n)}\left(\overline{\boldsymbol{x}}_m\right) \right) \tag{C.40}$$

En la Fig. C.1 (derecha), se muestra el OOB con puntos, los cuales indican que es una estimación precisa del error de detección.

1) Criterio de parada para *L*. Primero L = 1, y se va incrementando al ir decreciendo el error estimado de OOB de la ecuación (C.39). El valor de L no debe ser menor que 25 y mayor que 500. El entrenamiento se detiene una vez que las últimas K medias (C.41) calculadas a partir de μ de los errores OOB consecutivos superan el umbral épsilon ε. Esta condición se establece con las ecuaciones (C.40) y (C.41).

$$L = \ arg\min_{n}\left\{ n; \left| \min_{i\in P_K(n)} M_\mu(i) - \max_{i\in P_K(n)} M_\mu(i) \right| < \varepsilon \right\} \quad \text{(C.41)}$$

donde

$$M_\mu(i) = \frac{1}{\mu}\sum_{j=i-\mu+1}^{i} E_{OOB}^{(j)} \quad \text{(C.42)}$$

En $P_K(n) = \{n-k, \dots, n\}$. Los parámetros *K*, μ y ε son definidos por el usuario y controlan el equilibrio entre la complejidad computacional y la optimización. En este trabajo de investigación se utilizaron los valores recomendados por los autores, *K* = 50, $\mu = 5$ y $\varepsilon = 0.005$. Esto debido a los experimentos realizados por los autores, esta elección funciona universalmente para diferentes problemas de estegoanálisis.

2) Subespacio d_{sub}. Como se ilustra en la Figura C.1 (derecha) se tiene que buscar el mínimo valor de d_{sub}, ya que cuando la dimensión d_{sub} aumenta el nivel de clasificación tiende a decrecer. Para encontrar este parámetro, se utiliza el algoritmo de la Figura C.2. Se les asigna el valor a los parámetros de L1 (línea 1 del seudocódigo). En el primer escenario, se inicializa las variables *k*, E_{OOB} y d^*_{sub}. De L4 – L11 se tiene que calcular el error E_{OOB} con la dimensión $k \times \Delta d$, si el error e menor que E^*_{OOB} se almacena en esta última variable y también se guarda $k \times \Delta d$ en d^*_{sub}. Este ciclo se rompe cuando E_{OOB} es menor a la suma del umbral ε_d con E^*_{OOB}.

Siguiendo el seudocódigo de la Figura C.2. El segundo escenario es cuando el error empieza a incrementar. Entonces se tiene que regresar a una posición menor cuando el error era el mínimo. Para ello se redefine el incremento delata *d, a* $\Delta d/2$ hasta que se encuentre dentro de una tolerancia deseada τ. Los parámetros τ, Δd y ϵ_d controlan la compensación entre el tiempo de entrenamiento y la optimización de la solución. Los parámetros empleados en esta investigación, fueron los que recomiendan los autores, $\tau = 0.02$, $\Delta d = 200$, y $\epsilon_d = 0.005$.

Algorithm One-dimensional search for d_{sub}. To simplify the boundary issues, we define $E_{\text{OOB}}(d_{\text{sub}}) = 1$ for all $d_{\text{sub}} \notin [0, d]$.

Set parameters τ, Δ_d, and ϵ_d
//Stage 1: first pass with Δ_d
Initialize $k \leftarrow 0$, $E^*_{\text{OOB}} \leftarrow 1$, $d^*_{\text{sub}} \leftarrow 0$
repeat
    $k \leftarrow k + 1$
    Train ensemble classifier and obtain out-of-bag error estimate $E_{\text{OOB}}(k\Delta_d)$
    if $E_{\text{OOB}}(k\Delta_d) < E^*_{\text{OOB}}$ **then**
        $E^*_{\text{OOB}} \leftarrow E_{\text{OOB}}(k\Delta_d)$
        $d^*_{\text{sub}} \leftarrow k\Delta_d$
    end if
until $E_{\text{OOB}}(k\Delta_d) > E^*_{\text{OOB}} + \epsilon_d$
//Stage 2: localize the minimum by refining Δ_d
repeat
    Obtain $E_1 \equiv E_{\text{OOB}}(d^*_{\text{sub}} - \Delta_d)$
    Obtain $E_2 \equiv E_{\text{OOB}}(d^*_{\text{sub}})$
    Obtain $E_3 \equiv E_{\text{OOB}}(d^*_{\text{sub}} + \Delta_d)$
    if $1 \geq 2E_2/(E_1 + E_3) > 1 - \tau$ or Δ_d too small **then**
        return d^*_{sub}
    else
        $E^*_{\text{OOB}} \leftarrow \min\{E_1, E_2, E_3\}$
        $d^*_{\text{sub}} \leftarrow d_{\text{sub}}$ yielding E^*_{OOB}
        $\Delta_d \leftarrow \Delta_d/2$
    end if
until 1

Figura C.2. Algoritmo para calcular d_{sub}.

Anexo D. Entrenamiento y clasificación

Debido al problema que fue mencionado en la sección 5.3.4, se planteó una solución dividiendo la etapa de entrenamiento de la etapa de clasificación o prueba. En el código siguiente se detalla la implementación de la etapa de entrenamiento, se muestra en la Figura D.1 y D.2.

```
function tutorial_entrenamiento()
%--------------------------------------------------------------------------
% Última actualización Noviembre 2020 por Sergio Alexis R.V.
%--------------------------------------------------------------------------

% --------------------------------------------------------------------------
% Ensemble Classification | June 2013 | version 2.0 | TUTORIAL
% --------------------------------------------------------------------------
% Copyright (c) 2013 DDE Lab, Binghamton University, NY.
% All Rights Reserved.
% --------------------------------------------------------------------------
% Contact: jan@kodovsky.com | fridrich@binghamton.edu | June 2013
%          http://dde.binghamton.edu/download/ensemble
% --------------------------------------------------------------------------

% Short tutorial for the ensemble classifier (ver 2.0) developed for
% steganalysis in digital images.

% Cargar características.
cover = load('/ruta/cover.mat');
stego = load('/ruta/wow_050.mat');

% Ambos archivos tienen una matriz de características 'F', donde cada fila
% corresponde al un vector y el número de columnas es la dimencionalidad
% del vector. Contiene una estructura 'names' con el nombre de la imagen
% correspondiente al vector de características.

% Los nombres de stego/cover.names están desincronizados. La
% sincronizacion es importante como un paso de preprocesamiento en
% estegoanálisis. Po lo tanto, deben de sincronizarse en pares. Vector de
% imagen cover => vector imagen stego

%{
En el estegoanálisis, es importante entrenar en los * pares * de
características cover y las características  stego correspondientes.
Al dividir las características en partes de entrenamiento / prueba, por
ejemplo, estos pares deben conservarse. ¡Pero es igualmente importante
hacer un seguimiento de estos pares incluso dentro del conjunto de
entrenamiento! Esto se debe a que al dividir los datos de entrenamiento
con fines de validación cruzada (o bootstrapping), estos pares, nuevamente,
```

Figura D.1. Código para entrenamiento.

```
necesitan ser preservados. Por lo tanto, nuestra implementación de la
formación de conjunto acepta solo dos matrices * sincronizadas * de
características (cover y stego). Mientras que la implementación verifica
los tamaños de ambas matrices, la sincronización real es responsabilidad del
usuario. Vea el siguiente código como ejemplo de cómo hacer esto
correctamente.
%}
% Restricción: solo se consideran las características que tienen cover y su
% correspondientes stego, elimina los no pares c=>s
names = intersect(cover.names,stego.names); % cell array
names = sort(names); % ordena los nombres

% Preparar características cover C
% ismember(A,B) retorna vector A con 1 cuando es miembro B, 0 si no
cover_names = cover.names(ismember(cover.names,names));
% ix retona index de las posiciones iniciales
[cover_names,ix] = sort(cover_names);
% ismem retorna vector de 111..., obtienen los vectores
C = cover.F(ismember(cover.names,names),:);
C = C(ix,:); % aplica sort de ix

% Preparar características stego S
stego_names = stego.names(ismember(stego.names,names));
[stego_names,ix] = sort(stego_names);
S = stego.F(ismember(stego.names,names),:);
S = S(ix,:); % aplica sort de ix
%fprintf('size S:');
%disp(size(S))
%{
En este punto, tenemos las características cover C y las
características stego correspondientes S. Están sincronizadas
correctamente, es decir, la i-ésima fila de la matriz stego S proviene
de la imagen stego que se creó a partir de la imagen cover con
características en el i-ésima fila de la matriz cover C.
%}

tic
settings = struct('verbose',2);
for seed = 1:10
    RandStream.setGlobalStream(RandStream('mt19937ar','Seed',seed));
    random_permutation = randperm(size(C,1));% regresa vector aleatorio de N
    training_set = random_permutation(1:5000);%(1:round(size(C,1)/2));
    testing_set = random_permutation(5000+1:end);%(round(size(C,1)/2)+1:end);

    %guardando los vectores de prueba, cover y stego
    save(strcat('test_set',int2str(seed),'.mat'),'testing_set');

    TRN_cover = C(training_set,:); % devuelve matriz de vectores
    TRN_stego = S(training_set,:);

    % entrena el clasificador y regresa vector de pesos
    [trained_ensemble,results] = ensemble_training(TRN_cover,TRN_stego,settings);
    % guarda el entrenamiento
    save(strcat('trained_e',int2str(seed),'.mat'),'trained_ensemble','results');
    clearvars TRN_cover TRN_stego

end
toc
```

Figura D.2. Código para entrenamiento (continuación)

En la Figura D.3 y D.4 se detalla el código de la implementación de la etapa de clasificación.

```
function tutorial_prueba()
%--------------------------------------------------------------------------
% Última actualización Noviembre 2020 por Sergio Alexis R.V.
%--------------------------------------------------------------------------

% ------------------------------------------------------------------------
% Ensemble Classification | June 2013 | version 2.0 | TUTORIAL
% ------------------------------------------------------------------------
% Copyright (c) 2013 DDE Lab, Binghamton University, NY.
% All Rights Reserved.
% ------------------------------------------------------------------------
% Contact: jan@kodovsky.com | fridrich@binghamton.edu | June 2013
%          http://dde.binghamton.edu/download/ensemble
% ------------------------------------------------------------------------

% Short tutorial for the ensemble classifier (ver 2.0) developed for
% steganalysis in digital images.

% se cargan las características {F,names}
cover = load('/home/sergl/aletheia/Base/salida_mat/normal_cover.mat');
stego = load('/home/sergl/aletheia/Base/salida_mat/wow_050.mat');
% se asume que ambas caracteristicas, son simetricas

% Preparar características cover C
cover_names = cover.names;
% ix retorna index de las posiciones iniciales
[cover_names,ix] = sort(cover_names);

C = cover.F; % obtiene todos los vectores
C = C(ix,:); % aplica sort de ix

% Preparar características Stego S
stego_names = stego.names;
% ix retona index de las posiciones iniciales
[stego_names,ix] = sort(stego_names);

S = stego.F; % obtiene todos los vectores
S = S(ix,:); % aplica sort de ix

fprintf('Pruebas\n')
tic
%métricas
testing_errors = zeros(1,10);
accuracy_errors = zeros(1,10);
f_measure_erros = zeros(1,10);
for semilla = 1:10
    % lee caraterirticas para prueba (indices)
    TEST_ind = load(strcat('test_set',int2str(semilla),'.mat'));

    TEST_stego = S(TEST_ind.testing_set,:);
    TEST_cover = C(TEST_ind.testing_set,:);
    % lee el clasificador
    ENS = load(strcat('trained_e',int2str(semilla),'.mat'));
```

Figura D.3. Código para clasificación.

```
    test_results_cover = ensemble_testing(TEST_cover,ENS.trained_ensemble);
    test_results_stego = ensemble_testing(TEST_stego,ENS.trained_ensemble);

    % suma la cantidad de predicciones cuando son difererntes que -1
    % false positive FP
    false_alarms = sum(test_results_cover.predictions~=-1);
    % true negative TN
    true_negative = sum(test_results_cover.predictions~=+1);
    % false negative FN
    missed_detections = sum(test_results_stego.predictions~=+1);
    % true positive TP
    true_positive = sum(test_results_stego.predictions~=-1);
    % matriz de confusion
    fprintf('\n=====>  Stego | Cover \n');
    fprintf('Stego | %i  : %i\n',true_positive,false_alarms);
    fprintf('Cover | %i  : %i\n',missed_detections,true_negative);

    num_testing_samples = size(TEST_cover,1)+size(TEST_stego,1);
    % error PE
    testing_errors(semilla) = (false_alarms + missed_detections)/num_testing_samples;
    % accuracy
    accuracy_errors(semilla) = (true_positive + true_negative)/num_testing_samples;
    % F-measure
    f_measure_erros(semilla) =
(2*true_positive)/(2*true_positive+false_alarms+missed_detections);
    fprintf('\n Accuracy : %.4f\n',accuracy_errors(semilla));
    fprintf(' F-measure : %.4f\n',f_measure_erros(semilla));
    fprintf(' Testing error %i: %.4f\n',semilla,testing_errors(semilla));
    fprintf('---------------------------\n')
end
fprintf('\nAverage testing PE error over 10 splits: %.4f (+/-
%.4f)\n',mean(testing_errors),std(testing_errors));
fprintf('Average Accuracy error over 10 splits: %.4f (+/-
%.4f)\n',mean(accuracy_errors),std(accuracy_errors));
fprintf('Average F-measure error over 10 splits: %.4f (+/-
%.4f)\n',mean(f_measure_erros),std(f_measure_erros));
toc
clearvars
```

Figura D.4. Código para clasificación (continuación).