SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

SEP
Institutos Tecnológicos

Subsecretaría de Educación Superior
Dirección General de Educación Superior Tecnológica
Instituto Tecnológico de la Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Diseño e implementación de un control servo-visual de seguimiento de objetos en el robot movil Seekur"

POR

Ing. José Armando Sáenz Esqueda

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS

DR. Victor Adrian Santibañez Davila
M. en C. Edmundo Javier Ollervides Vázquez

ISSN: 0188-9060

TECNOLOGICO DE LA LAGUNA

RIITEC: (15)-TMCIE-2013

Torreón, Coahuila. México
Diciembre, 2013

"2013, Año de la Lealtad Institucional y Centenario del Ejercito Mexicano"

Dependencia: **DEPI**
Oficio: **DEPI/373/2013**
Asunto: **Autorización de impresión de tesis.**

Torreón, Coah., 07/Diciembre/2013

C. JOSE ARMANDO SAENZ ESQUEDA
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"Diseño e implementación de un control servo-visual de seguimiento de objetos en el robot móvil seekur"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro RITEC: **(15)-MCIE-2013**, para que proceda a la impresión del mismo.

ATENTAMENTE

DR. JOSE LUIS MEZA MEDINA
Jefe de la División de Estudios de Posgrado e Investigación

ESTADOS UNIDOS MEXICANOS
SECRETARIA DE EDUCACIÓN PUBLICA
INSTITUTO TECNOLÓGICO de la laguna
División de Estudios de Posgrado

"2013, Año de la Lealtad Institucional y Centenario del Ejército Mexicano"

Torreón, Coah., 06/Diciembre/2013

DR. JOSE LUIS MEZA MEDINA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"Diseño e implementación de un control servo-visual de seguimiento de objetos en el robot móvil seekur"

Desarrollado por el C.**JOSE ARMANDO SAENZ ESQUEDA**, con número de control **M06131436** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE

DR. VICTOR A. SANTIBAÑEZ DAVILA
Asesor/Director

M.C. EDMUNDO J. OLLERVIDES VAZQUEZ
Coasesor

DR. RICARDO E. CAMPA CÓCOM
Comité Tutorial

DR. MIGUEL ANGEL LLAMA LEAL
Comité Tutorial

Agradecimientos

Estas líneas son dedicadas a las personas que me ayudaron y estuvieron conmigo a lo largo de mis estudios de maestría.

A mis padres por toda su paciencia y esfuerzo, que en los momentos de mayor dificultad me dieron su apoyo a lo largo de esta etapa y nunca me dejaron caer.

A mi asesor Dr. Víctor Adrián Sanchibáñez Dávila, por escogerme como tesista y los conocimientos que me fueron impartidos en clase.

A mi co-asesor M. en C. Edmundo Javier Ollervides Vázquez, por todo el tiempo dedicado, sobre todo en la parte técnica de este trabajo.

A mis profesores de ingeniería y maestría, por los conocimientos adquiridos en clase.

Y por último, y no menos importante, al CONACYT, proyecto CONACYT 134534 y DGEST por la financiación y beca de este proyecto.

A todos ellos, muchas gracias.

Resumen

En este trabajo se presenta el diseño de un controlador para el robot móvil Seekur. Este robot puede ser configurado (u operado) ya sea como un robot omnidireccional sin restricciones no holonómicas o como un robot móvil tipo diferencial. Se presenta un modelo cinemático para el modo de operación omnidireccional y un modelo cinemático típico para el modo de operación diferencial. Se describe también cómo operar el robot empleando la biblioteca ARIA, la cual es proporcionada por el fabricante (MobileRobots). El objetivo de control es el seguimiento de un objeto empleando un sistema de visión con una cámara monocular. Se propone una ley de control tipo PD con saturación exponencial, en la cual se utiliza una técnica de control servo-visual basada en imagen. Finalmente se presentan los resultados experimentales obtenidos para el seguimiento de un objeto.

Abstract

This work presents the design of a controller for the Seekur mobile robot. This robot can be operated either as a omnidirectional robot without non-holonomic constraint or as a differential type mobile robot. A kinematic model for the omnidirectional mode and a typical kinematic model for the differential mode are presented. The document also describes how to operate the robot using the ARIA library, which is provided by the manufacturer (MobileRobots). The control aim is the tracking of a target by using a vision system with a monocular camera. By using an image based visual-servoing control technique, an exponential PD control law is proposed. Finally experimental results obtained for the tracking of an object are presented.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

Los robots están teniendo una gran aceptación, sobre todo en la sustitución del hombre por la máquina, en los trabajos que son repetitivos y por lo tanto tediosos, o trabajos denigrantes. En el área industrial, sobre todo, es posible encontrar líneas de producción totalmente automatizadas, siendo controladas por un pequeño grupo de personas, como lo son las líneas de ensamble, donde robots manipuladores con rutinas predefinidas producen automóviles en masa. Sin embargo otra clase de robots están comenzando a tener un gran auge, tanto dentro de la industria como en la vida cotidiana: son los robots móviles.

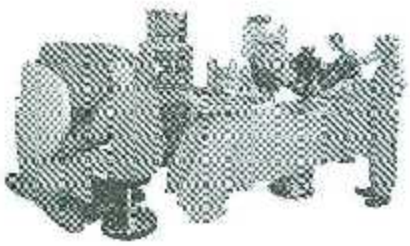

Figura 1.1: Robot DaVinci

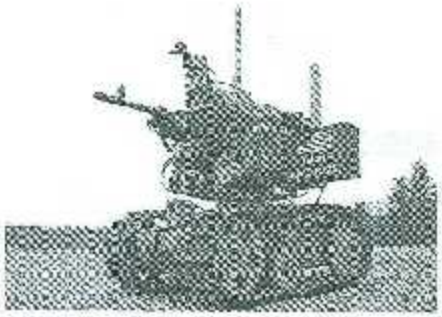

Figura 1.2: Robot militar TALON

Los robots manipuladores, ya pueden verse de manera común, en algunos restaurantes de Asia o incluso en los hospitales de todo el mundo, donde realizan operaciones quirúrgicas (cirugías) como el robot Da Vinci (véase Figura 1.1). Por otro lado, también existen robots

móviles como el robot TALON (véase Figura 1.2) que es de uso militar y se le pueden montar armas o brazos manipuladores. Para que los robots puedan ser capaces de realizar tareas de manera eficiente, hay que realizar extensos análisis sobre sus capacidades y limitaciones, principalmente en lo que se refiere a su diseño mecánico.

En la actualidad una de las áreas de control que está teniendo una gran aceptación es el control servo-visual. Con la tecnología actual es posible obtener una gran cantidad de información a través de un sensor de visión como lo es una cámara, la cual puede proporcionar dimensiones de objetos, texturas, colores, áreas, etc. Para procesar toda esta información se utilizan algoritmos muy complejos y matrices de gran tamaño, por lo que es necesario recurrir a computadoras con ciertas características que permitan procesar toda esta información en tiempo real. En la red, es posible encontrar una gran cantidad de algoritmos para procesar la información obtenida de una cámara. Una de las bibliotecas de software que se ha convertido en un pilar para procesamiento de imágenes es OpenCV, que está escrito en el lenguaje de programación C/C++ y es compatible con la mayoría de los sistemas operativos comerciales, tales como Windows y Linux.

Existen dos técnicas para aplicar visión en sistemas de control, la primera es llamada control servo-visual basado en imagen y la segunda control servo-visual basado en posición. La primera técnica tiene como objetivo de control que las características del objeto, previamente seleccionadas, como el centroide ó área converjan a características deseadas. El control servo-visual basado en imagen se realiza en el espacio de la imagen. Para alcanzar dicho objetivo el robot se mueve en el plano ignorando la posición y orientación a la que se encuentra el objeto. La segunda técnica recrea la imagen observada mediante una transformación para conocer la posición y orientación del robot respecto al objeto medido desde el marco del mundo. Los trabajos que utilizan control servo-visual son muy variados [Hauck et al., 2000, Yoshida and Tsuzuki, 2006, Ferrer, 1998, Copot et al., 2010, Espiau et al., 1992, Swain et al., 1998].

Para simular el comportamiento de un sistema con un controlador servovisual se pueden emplear diferentes modelos para conocer el comportamiento de una imagen. El primero es el modelo de cámara "pinhole", siendo este un modelo lineal, el cual considera que

la cámara funciona sin lente y la imagen es captada por un material fotosensible. El modelo de cámara pinhole es el empleado para esta investigación. El segundo modelo, conocido como de "lente delgada", es no lineal y, a diferencia del modelo anterior, considera que la proyección en el material fotosensible presenta una deformación. En la literatura existen una gran cantidad de modelos, siendo los dos mencionados los más utilizados. En [Mao et al., 2012, García, 2007, Siciliano et al., 2009] se encuentra la metodología para obtener el jacobiano de una imagen a partir de los modelos antes mencionados. El jacobiano de la imagen o matriz de interacción es empleada para conocer la evolución de la imagen a través del tiempo.

El robot utilizado es un robot móvil Seekur. Este robot es utilizado para exteriores y contiene una gran cantidad de sensores para distintos fines. Se puede clasificar como un robot móvil pseudo-omnidireccional porque cuenta con cuatro ruedas direccionables con orientación variable hasta 360° permitiéndole moverse en cualquier dirección y con cualquier orientación. La ventaja que presentan este tipo de robots móviles frente a los que utilizan ruedas suecas, que son los robots omnidireccionales, es la facilidad para conocer el punto de contacto de las ruedas con el suelo, porque el punto de contacto de las ruedas suecas varía con el movimiento de la rueda. Además el robot Seekur puede trabajar en un modo de operación diferencial. Algunos trabajos previos con el robot móvil Seekur son [Solea et al., 2010, Dumitrascu et al., 2011a, Dumitrascu et al., 2011b].

Las aplicaciones en la industria para el control servo-visual son muy variadas. Para el área de medicina se puede consultar [Kane et al., 2010], que muestra un control servo-visual en un robot móvil que está sujeto a un robot manipulador, este sistema sirve para realizar craneotomías con una mejor precisión. En exploración se encuentra el robot FinDER descrito en [Arroyo et al., 2012], su funcionamiento es en base a un sensor Kinect montado en la parte superior del robot el cual cuenta con sensores infrarrojos y una cámara RGB; esto le permite procesar las imágenes y recrear de manera eficiente un mapa 3D de la zona que está explorando. También se han hecho propuestas para el área de videovigilancia en [Chang et al., 2010], donde se emplea una cámara con dos grados de libertad (cámara PTZ, de pan/tilt/zoom) para seguir el movimiento de una persona con

ciertas características previamente especificadas.

El controlador propuesto en este trabajo es un controlador cinemático con una ley de control tipo PD con saturación exponencial. La ley de control propuesta contiene una exponencial con la finalidad de tener un comportamiento suave y saturado. El comportamiento de la ley de control es similar al de un controlador PD clásico saturado con la función tangente hiperbólica. Las funciones son muy parecidas mas no iguales, ya que el comportamiento de la exponencial es más lento respecto al de la tangente hiperbólica. Se demuestra que la función propuesta es diferenciable cuando menos una vez, empleándose límites para dicha demostración.

El controlador es probado en simulación para regulación y seguimiento, y de manera experimental únicamente en regulación. Para simulación se emplea la herramienta SIMULINK, tomando como modelo dinámico una aproximación lineal presentada en [Ollero, 2007] para el RAM-1 que tiene una configuración similar a la del robot Seekur. Para la experimentación se desarrolla el software en plataforma Linux Debian con el lenguaje de programación C/C++, las bibliotecas V4L2 y OpenCV para el procesamiento de imagenes, y ARIA para mandar las consignas de velocidad al robot Seekur.

1.1. Objetivos de la tesis

1.1.1. Objetivo general

- Desarrollar e implementar un control servo-visual basado en imagen para el seguimiento de objetos en el robot móvil Seekur, empleando una cámara monocular.

1.1.2. Objetivos específicos

- Proponer un controlador servo-visual cinemático con una ley de control PD.
- Desarrollar el software para procesamiento de imágenes y control.
- Realizar experimentos con el controlador propuesto.

1.2. Organización del documento

En el Capítulo 2 se presentan conceptos básicos sobre robots móviles y procesamiento de imagenes, así como el cálculo del jacobiano de una imagen. En el Capítulo 3 se hace una descripción del robot Seekur bajo estudio, en donde se indica la forma de operarlo y los distintos programas de computadora que pueden ser utilizados. El Capítulo 4 muestra dos modelos cinemáticos que corresponden a las formas de operación del robot (omnidireccional y diferencial), y el modelo cinemático de la cámara a emplear. El Capítulo 5 presenta el diseño del controlador tipo PD con saturación exponencial; también se muestran las simulaciones obtenidas empleando Simulink y los resultados obtenidos experimentalmente. Por último en el Capítulo 7 se dan las conclusiones del trabajo.

Capítulo 2

Marco teórico

2.1. Robots móviles con ruedas

2.1.1. Cinemática

Esta sección es tomada de [Canudas et al., 1996], donde se menciona que existen dos tipo de modelos cinemáticos para un robot móvil con ruedas. El primero es llamado *modelo cinemático de postura* en el cual se puede comprender las propiedades de movilidad del robot. El segundo es llamado *modelo cinemático de configuración* donde se analiza el robot desde el punto de vista de sistemas no holonómicos, obteniendo el modelo desde las restricciones no holonómicas de cada rueda. Ambos modelos consideran que el carro es un cuerpo rígido equipado con llantas no deformables con movimiento en un plano horizontal.

Para definir la postura se asignan marcos como se muestra en la figura 2.1 con respecto al marco G (X_g, Y_g) es el marco del mundo y el marco B (X_b, Y_b) es el marco móvil asignado al carro. Para describir la postura del robot se emplea el vector

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \tag{2.1}$$

donde las x, y son las coordenadas del marco móvil y θ es la rotación del marco B, ambos vistos desde el marco G. También existe una matriz de rotación

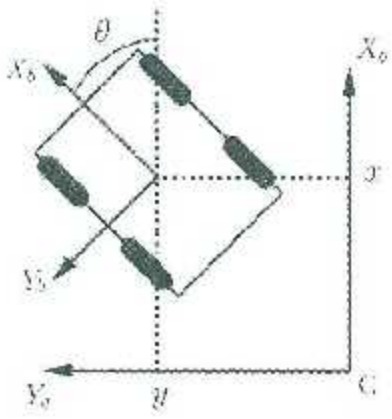

Figura 2.1: Asignación de marcos para un robot movil

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

que es la matriz de rotación que expresa la orientación del marco del mundo visto desde el marco móvil.

El movimiento del robot va de acuerdo al tipo de ruedas que se estén utilizando. Existen cuatro tipos de ruedas básicas que son: *fijas, orientables o direccionables, de castor y suecas.* Cada una de estas ruedas presenta restricciones no holonómicas. Estas restricciones sirven para conocer las trayectorias que pueden realizar las ruedas.

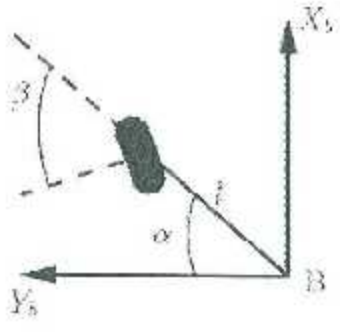

Figura 2.2: Rueda fija o rueda orientable

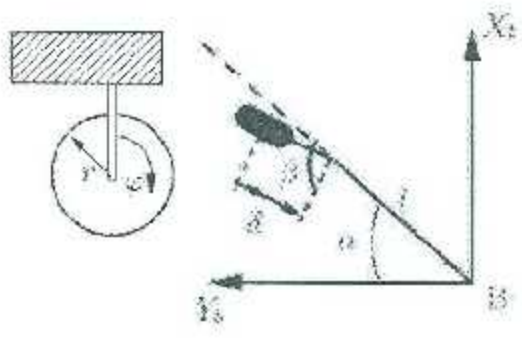

Figura 2.3: Rueda de castor

Las restricciones no holonómicas para cada una de las rueda son ([Canudas et al., 1996] y [Campion et al., 1996]):

- Rueda fija (véase Figura 2.2)

$$[-\sin(\alpha+\beta) \quad \cos(\alpha-\beta) \quad l\cos(\beta)]\, R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.3}$$

$$[\cos(\alpha-\beta) \quad \sin(\alpha+\beta) \quad l\sin(\beta)]\, R(\theta)\dot{\xi} = 0, \tag{2.4}$$

donde α, β y l describen la postura de la rueda respecto al marco del robot móvil, siendo las tres constantes; $\dot{\varphi}$ es la velocidad a la que gira la rueda y r es el radio de la rueda.

- Ruedas orientables (véase Figura 2.2)

$$[-\sin(\alpha+\beta) \quad \cos(\alpha-\beta) \quad l\cos(\beta)]\, R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.5}$$

$$[\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad l\sin(\beta)]\, R(\theta)\dot{\xi} = 0, \tag{2.6}$$

donde la representación es similar a la de la rueda fija, sólo que en una rueda orientable β es variable en el tiempo.

- Ruedas de castor (véase Figura 2.3)

$$[-\sin(\alpha+\beta) \quad \cos(\alpha-\beta) \quad l\cos(\beta)]\, R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.7}$$

$$[\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad d+l\sin(\beta)]\, R(\theta)\dot{\xi} + d\dot{\beta} = 0 \tag{2.8}$$

donde α, β, d y l describen la postura de la rueda respecto al robot móvil siendo α, l, d constantes y β varía en el tiempo, $\dot{\varphi}$ es la velocidad a la que gira la rueda y r es el radio de la rueda.

- Ruedas suecas (véase Figura 2.4)

$$[-\sin(\alpha-\beta+\gamma) \quad l\cos(\alpha-\beta+\gamma) \quad l\cos(\beta+\gamma)]\, R(\theta)\dot{\xi} + r\cos(\gamma)\dot{\varphi} = 0 \tag{2.9}$$

donde α, β, γ y l representan la postura de la rueda respecto al robot móvil, siendo constantes, $\dot{\varphi}$ es la velocidad a la que gira la rueda y r es el radio de la rueda.

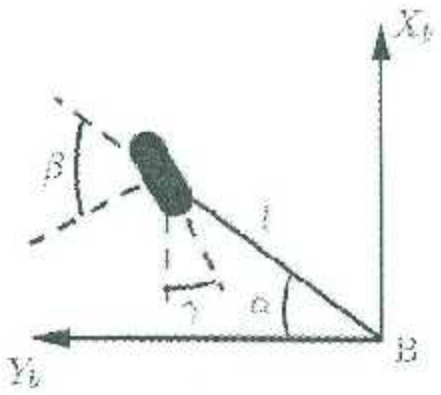

Figura 2.4: Rueda sueca

La configuración de un robot móvil puede ser descrita por los siguiente vectores:

- *coordenadas de postura del carro*; $\xi(t) = [x(t) \quad y(t) \quad \theta(t)]^T$ dan la postura del robot móvil en el plano,
- *coordenadas de orientación de las ruedas*: $\beta(t) = [\beta_s^T(t) \quad \beta_c^T(t)]^T$ dan los ángulos de orientación de las ruedas orientables (β_s) y la orientación de las ruedas de castor (β_c),
- *coordenadas de rotación de las ruedas* $\varphi(t) = [\varphi_f^T \quad \varphi_s^T \quad \varphi_c^T \quad \varphi_{sw}^T]^T$ dan los ángulos de rotación en el eje horizontal de las ruedas fijas (φ_f), ruedas orientables (φ_s), ruedas de castor (φ_c) y ruedas suecas (φ_{sw}).

Al conjunto de coordenadas de postura del carro, orientación y rotación de las ruedas se les llama *coordenadas de configuración*. El número de coordenadas de configuración se calcula con $N_f + 2N_s + 2N_c + N_{sw} + 3$, donde N_f es el número de ruedas fijas, N_s es el número de ruedas orientables, N_c es el número de ruedas de castor y N_{sw} es el número de ruedas de suecas.

Un robot móvil con ruedas se puede clasificar con la notación (δ_m, δ_s), siendo δ_m los grados de movilidad y el valor δ_s los grados de maniobrabilidad. La obtención de estos valores a partir de las restricciones no holonómicas de cada rueda se menciona en

[Campion et al., 1996], y tambien menciona que se pueden clasificar los robots móviles en cinco tipos básicos son:

- Tipo (3,0): robots móviles omnidireccionales.
- Tipo (2,0): robots móviles uniciclo.
- Tipo (2,1): robots móviles con una rueda orientable y dos de castor.
- Tipo (1,1): robots móviles tipo triciclo.
- Tipo (1,2): robots con 2 ruedas orientables y una de castor.

Existen una gran cantidad de robots móviles con ruedas, por lo que la lista anterior sólo menciona algunos de los muchos posibles robots que se pueden clasificar de esta manera.

2.2. Procesamiento de imagenes

Cuando se emplea control servo-visual es necesario extraer de la imagen adquirida por la cámara la información que sea útil para el controlador. En [Siciliano et al., 2009] se menciona que el procesamiento de la imagen previo a la realimentación consta de dos etapas: segmentación e interpretación.

La definición de una imagen escrita en [Cuevas et al., 2010] es: "Una imagen puede definirse como una función bidimensional que cuantifica la intensidad de luz". La imagen se representa de acuerdo al modelo de color que se utilice, siendo el modelo más común el RGB. El modelo RGB utiliza tres valores para cada punto de la imagen, que representan la intensidad de rojo, azul y verde.

2.2.1. Segmentación

Segmentación significa separar una imagen en regiones u objetos constituyentes. El proceso de segmentación finaliza al momento que los objetos de interés en una imagen han sido aislados como se muestra en la Figura 2.5.

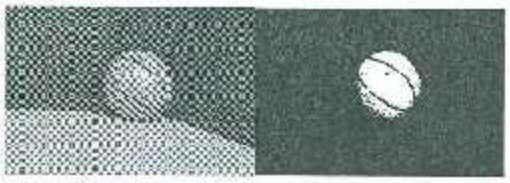

Figura 2.5: Resultados de la segmentación

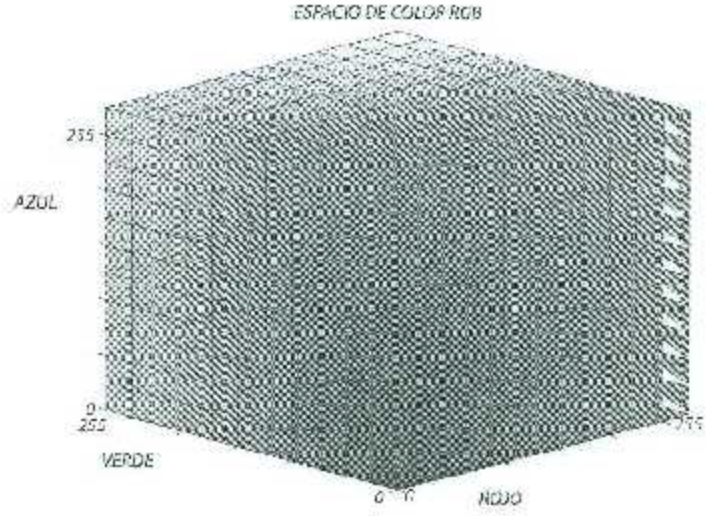

Figura 2.6: Espectro de colores en formato RGB

Usualmente estos procedimientos utilizan los valores de intensidad de la imagen identificando en donde se encuentra una discontinuidad, como se indica a continuación:

- Particionar una imagen en base a fuertes cambios en el valor de la intensidad (tales como los bordes).
- Particionar una imagen en regiones que son similares de acuerdo a un conjunto de criterios predefinidos.

La etapa de segmentación se basa en la detección de los tres tipos básicos de discontinuidades que son: nivel de gris, puntos y bordes. Al momento en que se obtiene una imagen, esta se almacena en memoria con un formato RGB (Red-Green-Blue, Rojo-Verde-Azul). El espacio de color RGB se muestra en la Figura 2.6. Se crean tres matrices $I_R, I_V, I_A \in \mathbb{R}^{n \times m}$

donde n y m son números enteros dados por la resolución de la cámara (cantidad de pixeles de la imagen captada por la cámara), y los elementos i, j de las matrices I_R, I_V e I_A denotados por I_{Ri}, I_{Vij} e I_{Aij} con $i = 1, 2, ..., n$, $j = 1, 2, ..., m$ son enteros que toman valores entre 0 y 255, es decir, I_{Rij}, I_{Vij} e $I_{Aij} \in [0, 255]$. Cada elemento de las tres matrices antes mencionadas indican la intensidad de rojo, verde y azul, respectivamente, para cada pixel presente en una imagen.

El primer paso de la segmentación es eliminar la mayor cantidad de información no necesaria en la imagen. Esto se realiza empleando un filtro de color donde cada elemento de las matrices I_R, I_V e I_A es acotado por arriba y por abajo por un umbral seleccionado de acuerdo a la intensidad de rojo, verde y azul, según el color deseado. En caso de que el valor se salga de la cota se le asigna un cero al elemento. El resultado de esta operación se observa en la Figura 2.7, donde las nuevas matrices resultantes del filtro de color son denotadas por I_{RC}, I_{VC} e I_{AC}.

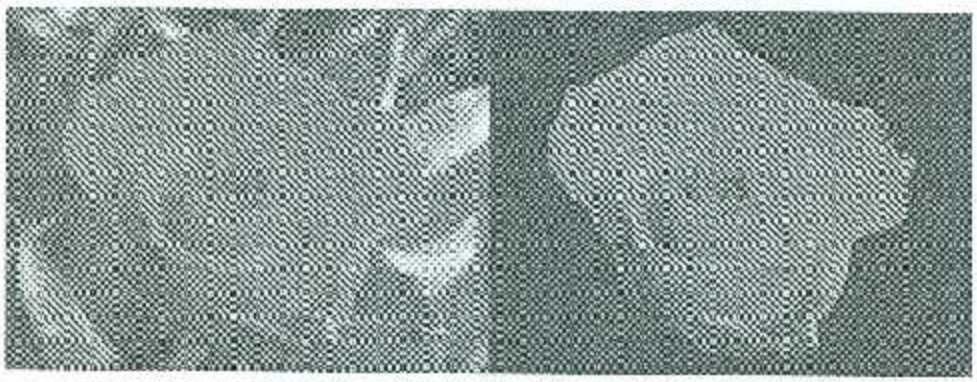

Figura 2.7: Resultado de un filtro de color

Después de realizar el filtraje de color se pasa a crear una imagen binaria. La imagen binaria se crea con una matriz cuyos elementos son 1 ó 0. Para generar la imagen binaria se van a emplear las matrices de la imagen obtenida después de aplicar el filtro de color I_{RC}, I_{VC} e I_{AC}. A partir de estas tres matrices se crea una nueva matriz I_B de la misma dimensión que las anteriores, donde el valor del elemento $[i, j]$ es cero si el mismo elemento correspondiente a las matrices I_{RC}, I_{VC} y I_{AC} es igual a cero; en cambio el valor del elemento será 1 si cuando menos un elemento de las matrices I_{RC}, I_{VC} y I_{AC} es diferente

de cero. La imagen binaria obtenida a partir de la Figura 2.7 es mostrada en la Figura 2.8.

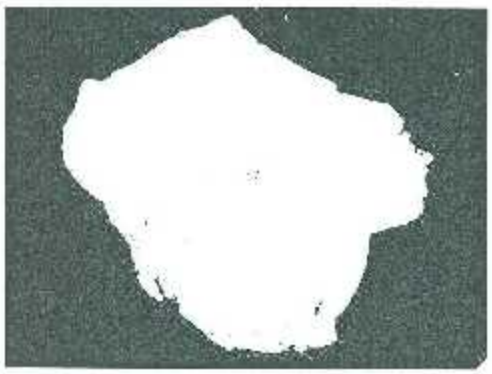

Figura 2.8: Imagen binaria

Para tener una imagen más clara del objeto que se piensa identificar se aplica la operación "erosión" a la imagen binaria. Esta operación se aplica a matrices y es representada con el simbolo $\ominus$, donde la matriz que se coloca a la izquierda del simbolo es la matriz de la imagen binaria I_B y la matriz que está al lado derecho es la matriz M, llamada máscara, con el patrón que se desea identificar. Esta operación de erosión sirve para filtrar información que no es posible eliminar mediante el filtrado de color, como lo son objetos con el mismo tipo de color pero no con el mismo tipo de forma. Para aplicar la operación de erosión es necesario utilizar una máscara que permite comparar las características actuales de la imagen con las características que se desean identificar. Una máscara es una matriz $M \in \mathrm{B}^{l \times s}$ donde $\mathrm{B} = \{0, 1\}$ de una dimensión inferior que la matriz obtenida por la imagen cuya dimensión es definida por el diseñador. Si la máscara es grande el tiempo de procesamiento es corto, pero la imagen puede contener información innecesaria; en cambio, si la máscara es pequeña la información tiene un mejor filtrado, pero el tiempo de procesamiento es mayor. El resultado obtenido al aplicar la operación de erosión es una matriz de imagen M_E del mismo tamaño que la original, es decir de $n \times m$ con los detalles bien definidos. El conjunto de números que se ingresen en cada uno de los elementos de la máscara debe de pertenecer al conjunto de números que aparezcan en la imagen original. En el caso de tener una imagen como la mostrada en la Figura 2.8, la matriz que se obtiene

es binaria, por lo tanto la máscara que se aplica debe de contener datos binarios. La forma de realizar la operación de erosión es ir comparando la máscara con cada submatriz del mismo tamaño encontrada para cada elemento en la imagen; si concuerdan los elementos de las submatrices de la imagen binaria con los de la máscara, el valor de ese elemento es 1, si no concuerda es 0 como se muestra a continuación:

$$I_B \otimes M = M_E$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

El resultado de una operación de erosión en una imagen binaria M_E es ilustrado en la Figura 2.9.

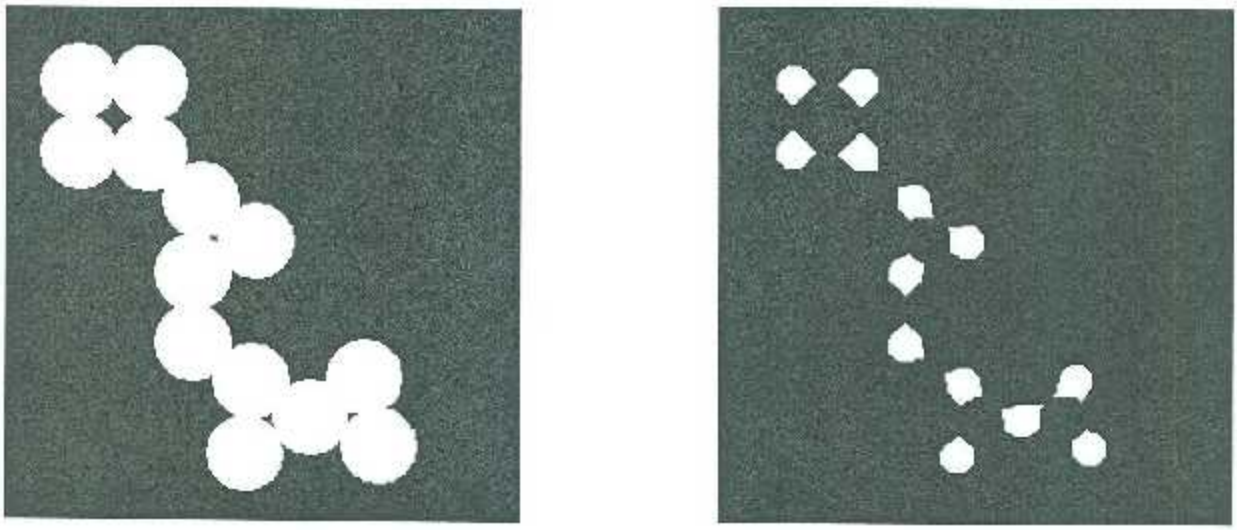

Figura 2.9: Resultados de una operación de erosión

2.2.2. Interpretación

La interpretación de una imagen es el proceso en el cual los cálculos realizados indican las características del objeto observado a partir de los segmentos obtenidos en el proceso

anterior. Estas características pueden ser límites o regiones. Las características son obtenidas para aplicaciones de control servo-visual y éstos a veces requieren del cálculo de los llamados momentos. Los momentos de una imagen se definen en una región R de la imagen y pueden ser utilizados para caracterizar la posición, la orientación y la forma del objeto bidimensional correspondiente a la región.

La expresión de un momento $m_{i,j}$ de una región R de una imagen observada con $\{i, j\} \in \mathbb{N}$, es la siguiente:

$$m_{i,j} = \sum_{X_I Y_I \in R} I(X_I, Y_I) X_I^i Y_I^j$$

donde X_I y Y_I representan las coordenadas de los pixeles, e $I(X_I, Y_I) \in \mathbb{R}^{w,h}$ con $w, h \in \mathbb{N}$, es la función de intensidad de la imagen. En el caso de imágenes binarias, suponiendo que la intensidad de la luz es igual a uno para todos los puntos de la región R y es igual a cero para aquellos puntos no pertenecientes a R, se obtiene la siguiente definición simplificada de momento:

$$m_{i,j} = \sum_{X_I Y_I \in R} X_I^i Y_I^j \tag{2.10}$$

En vista de esta definición, el momento $m_{i,0}$ coincide con el área de la región, calculada en términos de la cantidad total de pixeles de la región R. Las expresiones

$$\bar{x} = x_1 = \frac{m_{1,0}}{m_{0,0}} \tag{2.11}$$

$$\bar{y} = x_2 = \frac{m_{0,1}}{m_{0,0}}, \tag{2.12}$$

sirven para calcular las coordenadas del centroide de la región. Estas coordenadas se pueden utilizar para detectar exclusivamente la posición de la región R en el plano de la imagen. Usando una analogía de la mecánica, la región R puede ser vista como un cuerpo rígido bidimensional de densidad igual a la intensidad de la luz. Por lo tanto, el momento $m_{0,0}$ corresponde a la masa del cuerpo y el centroide corresponde al centro de masa.

El valor del momento $m_{i,j}$ en (2.10) depende de la posición de la región R en el plano de la imagen. Por lo tanto los llamados momentos centrales son definidos como

$$\mu_{i,j} = \sum_{X_I Y_I \in \mathcal{R}} (X_I - \bar{x})^i (Y_I - \bar{y})^j \tag{2.13}$$

los cuales son invariantes con respecto a la traslación.

De acuerdo con la analogía mecánica, es fácil reconocer que los momentos centrales de segundo orden $\mu_{2,0}$ y $\mu_{0,2}$ tienen el significado de momentos de inercia con respecto a los ejes X_I y Y_I, respectivamente, mientras que $\mu_{1,1}$ es un producto de inercia, y la matriz

$$\Gamma = \begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix},$$

tiene el significado de tensor de inercia con respecto al centro de la masa. Los valores propios de la matriz Γ son los momentos principales de inercia, denominados momentos principales de la región y los vectores propios correspondientes definen los ejes principales de inercia, denominados ejes principales de la región.

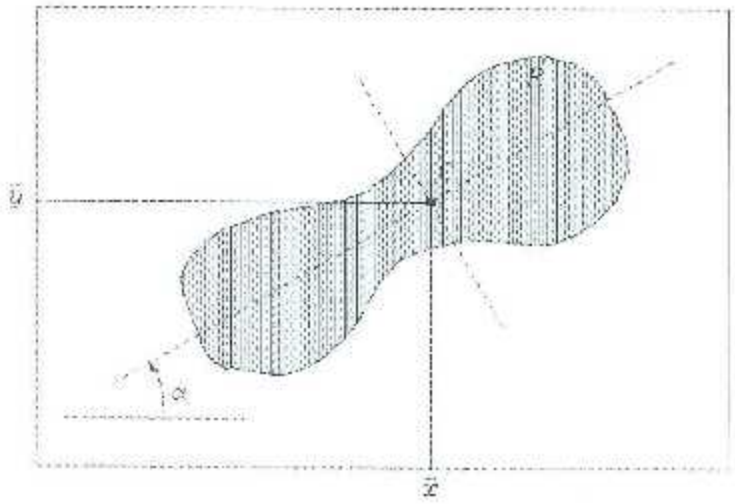

Figura 2.10: Región de una imagen binaria

Si la región R es asimétrica, los momentos principales de Γ son diferentes y es posible definir la orientación de R en términos del ángulo α que se encuentra entre el eje principal correspondiente al momento máximo y el eje X. Este ángulo se puede calcular con la siguiente expresión

$$\alpha = \frac{1}{2} \arctan\left(\frac{\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right)$$

Como un ejemplo, en la Figura 2.10, se muestra la región de una imagen binaria con un centroide C, los ejes principales y el ángulo α. Note que los momentos y los parámetros correspondientes también pueden ser calculados a partir de las fronteras de los objetos, además, estas cantidades son especialmente útiles para caracterizar objetos de forma genérica. A menudo, sin embargo, la objetos presentes en la escena, especialmente los fabricados, tienen características geométricas que son útiles para la interpretación de las imágenes. Por ejemplo, muchos objetos tienen bordes que, en el plano de la imagen, corresponden a la intersección de las partes lineales de contorno o de puntos de contorno de alta curvatura. Las coordenadas de estos puntos en el plano de la imagen se pueden detectar utilizando algoritmos robustos para rechazar el ruido, y por lo tanto se pueden utilizar como características de la imagen. Por lo general, se denominan "puntos característicos" (feature points).

En otros casos, es posible identificar primitivas geométricas verdaderas tales como líneas o segmentos de línea, que son proyecciones de bordes lineales o sólidos de revolución (conos, cilindros o elipses), obtenidos como proyecciones de círculos o esferas. Estas primitivas pueden ser caracterizadas en el plano de la imagen en términos de una conjunto mínimo de parámetros. Por ejemplo, un segmento de línea se puede caracterizar por las coordenadas de sus puntos finales, o alternativamente, por las coordenadas de su punto medio (centroide), su longitud (momento m_{00}) y su orientación (ángulo α). En ambos casos, la caracterización del segmento de línea requiere cuatro parámetros.

2.3. Jacobiano de una imagen

La matriz de interacción o jacobiano de una imagen permite conocer las velocidades de las características de un objeto tridimensional en un plano definido por el sensor de la cámara. La matriz de interacción a utilizar parte de un modelo de cámara tipo pinhole. El tipo de cámaras pinhole considera cámaras sin lente y con una pequeña apertura por la cual entra la luz que es captada por un material fotosensible. La metodología empleada en este trabajo para la obtención del jacobiano de la imagen, fue obtenida de [García, 2007].

En [Mao et al., 2012] se presenta una nueva forma de obtener el jacobiano con estimaciones en línea. La matriz pertenece al espacio $\mathbb{R}^{2m\times 6}$ donde m es el número de puntos a seguir.

Para comprender la utilidad del jacobiano de la imagen, supóngase que se dispone de un sistema de cámara fija que observa el movimiento de un objeto cualquiera. A partir de las imágenes capturadas, el sistema de visión es capaz de extraer un punto característico, $\boldsymbol{p}_e^c$, los subindices $_e^c$ indican que es un punto en el espacio (e) visto desde el marco de la cámara (c). En un instante de tiempo $\boldsymbol{p}_e^c$ se moverá con una velocidad lineal $\boldsymbol{v}_e^c = [\dot{x}_e^c \quad \dot{y}_e^c \quad \dot{z}_e^c]^T$ así como con una velocidad angular $\boldsymbol{\omega}_e^c = [\omega_{xe}^c \quad \omega_{ye}^c \quad \omega_{ze}^c]^T$ ambas con respecto al sistema de coordenadas de la cámara, donde $\omega_{xe}^c, \omega_{ye}^c, \omega_{ze}^c$ indican la rotación alrededor de los ejes x, y y z, respectivamente. A su vez, el sistema de visión captará la evolución temporal de esta característica representada por $\boldsymbol{s}$, en el plano de la imágen $(\dot{x}^s, \dot{y}^s)$. El jacobiano de la imagen, L, es una matriz que relaciona las velocidades de las características en el plano de la imagen con las velocidades lineal y angular de los puntos correspondientes a las características, con respecto al sistema de coordenadas de la cámara. De esta forma se puede conocer cómo cambian las características en el plano de la imagen cuando se produce un cambio en el extremo del robot en movimiento:

$$\begin{bmatrix} \dot{x}^s \\ \dot{y}^s \end{bmatrix} = L(\boldsymbol{p}_e^c) \begin{bmatrix} \boldsymbol{v}_e^c \\ \boldsymbol{\omega}_e^c \end{bmatrix} \tag{2.14}$$

Ahora considerese que se va a representar con $\boldsymbol{s}$ un vector de i características observadas en la imagen, mientras que $\dot{\boldsymbol{s}}$ será el flujo óptico o variación de estas características en la imagen. El jacobiano L realiza la siguiente transformación:

$$\begin{bmatrix} \dot{x}_1^s \\ \dot{y}_1^s \\ \vdots \\ \dot{x}_i^s \\ \dot{y}_i^s \end{bmatrix} = \begin{bmatrix} L_1(\boldsymbol{p}_e^c) \\ \vdots \\ L_i(\boldsymbol{p}_e^c) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_e^c \\ \boldsymbol{\omega}_e^c \end{bmatrix} \tag{2.15}$$

Como se observa en la expresión (2.15), el número de columnas en el Jacobiano de la imagen variará dependiendo de la tarea ($j = 6$). En general la matriz L tendrá $2i$ filas y 6 columnas siendo cuadrada únicamente cuando se considere identificar 3 puntos de las imagen.

A continuación se va a determinar de forma teórica el valor del jacobiano de la imagen. Para ello se parte del modelo de cámara "pinhole". Sea un punto p^G de la escena 3D en coordenadas del mundo $p^G = [x_p^G \quad y_p^G \quad z_p^G]$ y siendo $p^S = [x_p^S \quad y_p^S]$ su proyección en el plano imagen (véase Figura 2.11), la proyección p^S en el plano imagen del punto p^G se obtiene como intersección de la línea que une p^G y el centro óptico de la cámara C con el plano imagen. La distancia entre el centro óptico C y el plano imagen se denomina distancia focal f. Se llama punto principal al punto donde el eje z^c intercepta al plano de la imagen, siendo este punto donde se asigna el marco de la imagen.

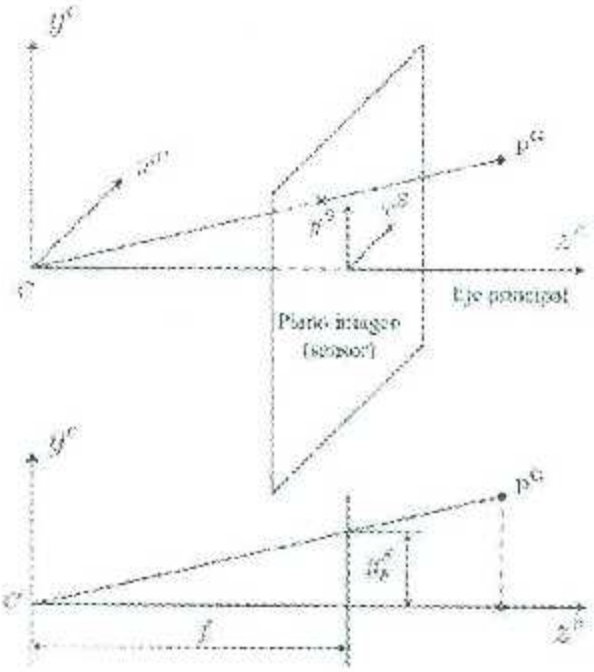

Figura 2.11: Definición de marcos para un modelo de cámara pinhole

En el modelo de cámara "pinhole", el sistema de lentes de la cámara es aproximado por un único punto. Con esta aproximación la imagen siempre se encuentra enfocada sin importar la distancia entre el objeto y las lentes, o la distancia entre las lentes y el plano imagen. De esta manera, cuando se utiliza el modelo "pinhole" un punto en el sistema de coordenadas 3-D de la cámara, C, se proyecta en un punto en el sistema de coordenadas

2-D del sensor, S, de la siguiente manera:

$$\boldsymbol{p}^S = \frac{f}{z^c}\boldsymbol{p}^c \tag{2.16}$$

donde f es la distancia focal de la cámara y z^c es la distancia de $\boldsymbol{p}^C$ a lo largo del eje principal.

Derivando con respecto al tiempo la expresión (2.16), se obtiene:

$$\dot{\boldsymbol{p}}^S = \frac{-f\dot{z}^c}{(z^c)^2}\boldsymbol{p}^c + \frac{f}{z^c}\dot{\boldsymbol{p}}^c \tag{2.17}$$

Considerando una cámara en movimiento y un objeto fijo, la ecuación fundamental de la cinemática toma el siguiente valor [Baer et al., 2007]:

$$\dot{\boldsymbol{p}}^c = \begin{bmatrix} \dot{x}^c \\ \dot{y}^c \\ \dot{z}^c \end{bmatrix} = \boldsymbol{\omega}_c^c \times \boldsymbol{p}^c - \boldsymbol{v}_c^c \tag{2.18}$$

donde $\boldsymbol{\omega}_c^c$ es la velocidad angular de la camara y $\boldsymbol{v}_c^c$ es la velocidad lineal de la cámara. Desarrollando la expresión anterior se tiene:

$$\dot{\boldsymbol{p}}^c = \begin{bmatrix} \omega_{xc}^c \\ \omega_{yc}^c \\ \omega_{zc}^c \end{bmatrix} \times \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} + \begin{bmatrix} \dot{x}_c^c \\ \dot{y}_c^c \\ \dot{z}_c^c \end{bmatrix} = \begin{bmatrix} \dot{x}_c^c - y^c\omega_{zc}^c - z^c\omega_{yc}^c \\ \dot{y}_c^c - z^c\omega_{xc}^c + x^c\omega_{zc}^c \\ \dot{z}_c^c - x^c\omega_{yc}^c + y^c\omega_{xc}^c \end{bmatrix}. \tag{2.19}$$

Sustituyendo la ecuación (2.19) en la expresión (2.17), se obtiene:

$$\dot{\boldsymbol{p}}^S = -f\frac{\dot{z}_c^c - x^c\omega_{yc}^c + y^c\omega_{xc}^c}{(z^c)^2}\begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} + \frac{f}{z^c}\begin{bmatrix} \dot{x}_c^c - y^c\omega_{zc}^c + z^c\omega_{yc}^c \\ \dot{y}_c^c - z^c\omega_{xc}^c - x^c\omega_{zc}^c \\ \dot{z}_c^c - x^c\omega_{yc}^c - y^c\omega_{xc}^c \end{bmatrix}, \tag{2.20}$$

que si se desarrolla queda:

$$\dot{\boldsymbol{p}}^S = \begin{bmatrix} \frac{f\dot{x}_c^c}{z^c} - \frac{x^c\dot{z}_c^c f}{(z^c)^2} - \frac{x^c y^c\omega_{zc}^c f}{(z^c)^2} + \left(f + \frac{(x^c)^2 f}{(z^c)^2}\right)\omega_{yc}^c - \frac{y^c\omega_{xc}^c f}{z^c} \\ \frac{f\dot{y}_c^c}{z^c} - \frac{y^c\dot{z}_c^c f}{(z^c)^2} + \left(f - \frac{(y^c)^2 f}{(z^c)^2}\right)\omega_{xc}^c + \frac{x^c y^c\omega_{yc}^c f}{(z^c)^2} - \frac{x^c\omega_{zc}^c f}{z^c} \\ 0 \end{bmatrix}. \tag{2.21}$$

Sustituyendo las ecuaciones correspondientes al modelo de cámara "pinhole",

$$x^S = \frac{f}{z^c}x^c \qquad y^S = \frac{f}{z^c}y^c, \tag{2.22}$$

en la ecuación (2.21) se llega a

$$\begin{bmatrix} \dot{x}^S \\ \dot{y}^S \end{bmatrix} = \begin{bmatrix} \frac{f\dot{x}^c}{z^c} - \frac{x^S\dot{z}^c}{z^c} - \frac{x^Sy^S\omega^c_{xc}}{f} + \left(f + \frac{(x^S)^2}{f}\right)\omega^c_{yc} - y^S\omega^c_{zc} \\ \frac{f\dot{y}^c}{z^c} - \frac{y^S\dot{z}^c}{z^c} - \left(f + \frac{(y^S)^2}{f}\right)\omega^c_{xc} + \frac{x^Sy^S\omega^c_{yc}}{f} + x^S\omega^c_{zc} \end{bmatrix}. \tag{2.23}$$

Con lo que se obtiene la siguiente relación en la que se observa el valor del jacobiano de la imagen

$$\begin{bmatrix} \dot{x}^S \\ \dot{y}^S \end{bmatrix} = \begin{bmatrix} \frac{f}{z^c} & 0 & \frac{x^S}{z^c} & \frac{x^Sy^S}{f} & \frac{f^2+(x^S)^2}{f} & y^S \\ 0 & -\frac{f}{z^c} & \frac{y^S}{z^c} & \frac{f^2+y^S}{f} & -\frac{x^Sy^S}{f} & -x^S \end{bmatrix} \begin{bmatrix} \dot{x}^c_c \\ \dot{y}^c_c \\ \dot{z}^c_c \\ \omega^c_{xc} \\ \omega^c_{yc} \\ \omega^c_{zc} \end{bmatrix} \tag{2.24}$$

Capítulo 3

Robot Seekur

El robot a utilizar es un robot móvil Seekur de la empresa "Mobile Robots" (véase Figura 3.1). Otros robots con una configuración de ruedas similar al robot Seekur son los robots Care-O-bot 3 [Connette et al., 2008] y AZIMUT-3 [Clavien and Fr, 2010]. Los robots móviles pseudo omnidireccionales son robots de cuatro ruedas orientable, teniendo la capacidad de poder desplazarse en el plano sin importar su orientación. Un robot pseudo-omnidireccional para hacer un cambio en su trayectoria hay que esperar a un cambio de postura de sus ruedas, mientras que en un robot omnidireccional es posible cambiar de trayectoria sin esperar a la reorientación de sus ruedas.

El Seekur es un robot especialmente diseñado para fines de investigación, con capacidad de operar en exteriores, por lo que contiene una gran cantidad de sensores y actuadores para esta finalidad; además es posible agregarle más accesorios como lo son cámaras o robots manipuladores. Seekur incluye software para distintos fines que pueden ser teleoperación, creación de rutinas, simulaciones de rutas, mapeo de una zona, adquisición de imágenes, entre otras. En las siguientes secciones se mencionan las características del robot, parámetros y software.

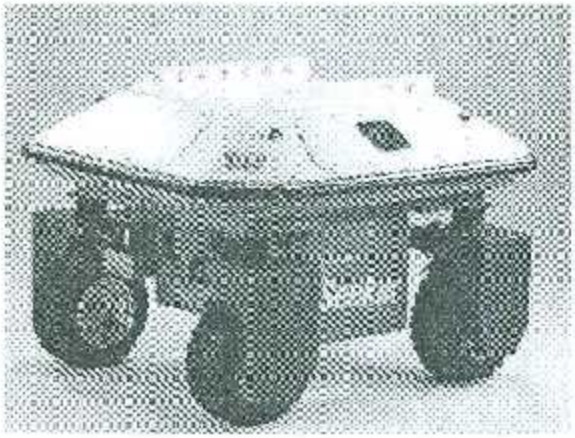

Figura 3.1: Robot Seekur

3.1. Descripción del robot

Este robot está diseñado para trabajos en exteriores por lo que presenta ciertas características para desempeño en espacios abiertos. La parte externa del robot presenta placas de aluminio con una parte desmontable en cada uno de sus cuatro lados; esta parte está sujeta por tornillos y en la parte interior tiene un empaque para su correcto sellado impidiendo el paso de líquidos al interior del robot. Para la suspensión tienen un resorte en la articulación que une la llanta con el cuerpo del robot y como amortiguación se tiene un lubricante de alta densidad, el cuál puede observarse en la articulación que une la llanta al robot. El sistema de transmisión para la tracción, que es un sistema de engranes en cada una de las llantas, se encuentra protegido por placas de aluminio y la transmisión del direccionamiento de las llantas, que está dentro del robot, es una transmisión de poleas y engranes. Cada llanta tiene un motor para su tracción y otro para su dirección. Los motores para la tracción no se encuentran protegidos por lo cual son susceptibles a recibir daño físico mientras que los motores para el direccionamiento se encuentran dentro del motor.

Para conocer el desplazamiento del robot tomando como origen el lugar donde se encendió, la medición se hace empleando codificadores ópticos incrementables o con lecturas del GPS. Para la estimación del desplazamiento se cuenta con codificadores ópticos incre-

mentables tanto en los motores de tracción como en el motor de direccionamiento, además de un giroscopio de eje único, el cual sirve para conocer la orientación del robot. También cuenta con un centro de acelerómetros el cual tiene la finalidad de estimar la velocidad que tiene el robot en cada instante de tiempo. Como una segunda opción se cuenta con un GPS marca Trimble, el cual presenta un margen de error de 2 metros. Ambos métodos entregan lecturas de desplazamiento y velocidad.

Para dar órdenes al robot o tomar las lecturas se utiliza una computadora a bordo. El sistema operativo que está pre-instalado es "Linux Debian 5.0 " o "Linux Lenny". El robot opera con un microprocesador que se encuentra justo debajo de la computadora, el cual recibe y manda información a través de un puerto serial que está conectado a la computadora. El puerto que se utiliza para establecer esta comunicación es "COM1". Para poder accesar a esta computadora se tiene que retirar la placa que se encuentra en el lado izquierdo del robot quedando al descubierto cuatro puertos seriales, un puerto PS2 para el teclado, un puerto PS2 para mouse, dos puertos para USB y un puerto VGA para lo cual resta conectar los dispositivos necesarios (monitor, teclado USB o PS2 y un mouse USB o PS2) para una comunicación local. La cuenta de usuario es "root" y la contraseña es "mobilerobots"

Es posible establecer también una comunicación remota que se realiza por medio de protocolo WiFi a tráves de un ruteador que se encuentra en la parte superior del robot y está cubierto por una caja de plástico sellada herméticamente para ser usado a la intemperie. Este ruteador tiene instalado el paquete "Monowall" que es un cortafuegos y se configura para permitir que el ruteador sea visto o no por todos los dispositivos que cuenten con una tarjeta inalámbrica, así como también restringir los IP's o direcciones mac que puede conectarse. La computadora a bordo cuenta con una IP fija que es "10.0.125.32". Para establecer la comunicación con el microprocesador es necesario correr los programas ARNL o MOGS que convierten la computadora a bordo en un servidor. Para ejecutar programas de manera remota se requiere de los paquetes "SSH" si se usa Linux en el cliente o "Putty" en caso de Windows. El usuario y contraseña son los mismos que para acceso local.

Para el mapeo del entorno se emplea el LRF (Laser Range Finder) que se encuentra en la parte frontal del robot. Este sensor se encuentra a una altura de 58 cm por lo tanto todo objeto que esté por debajo de esta altura o presente huecos a esta altura no es correctamente detectado. La información obtenida puede ser guardada en un archivo de extension ".map", el cual puede ser visto posteriormente con el paquete "Mapper3" para recrear un mapa de la zona.

Las cámaras que están instaladas en el robot son para visión monocular y visión estereo. Para la visión monocular se cuenta con una cámara PTZ marca RVision modelo SEE HP; esta cámara es analógica por lo cuál cuenta con una adquisitora de imagenes (framegrabber) Sensoray 2253 para digitalizar las imágenes, la cual se conecta por puerto USB con la computadora a borco. Para adquirir las imágenes de este dispositivo es necesario el paquete "V4L2" que viene preinstalado. Para visión estereo se emplea la cámara MobileRanger C3D que requiere una tarjeta Focus Robotics nDepth con interfaz PC104. Todos estos dispositivos se muestran en la Figura 3.2. Para la adquisición de imágenes se usan funciones de la biblioteca V4L2 para establecer la comunicación y el procesamiento de imagenes se emplea la biblioteca OpenCV

El robot trabaja con un módulo de baterias que se encuentran en un compartimento en la parte inferior del robot, este compartimento está pintado de gris y tiene retulado el logo del fabricante. El módulo contiene diferentes tipo de baterías, cada una con un proposito diferente; algunas son para los sensores y computadora a bordo, mientras que otras son para los actuadores. Las baterías que se emplean son:

- 2 baterías de 5 VDC/ 2 A, reguladas
- 4 baterías de 12 VDC/ 2 A, reguladas
- 2 baterías de 24 VDC/ 1 A, reguladas
- 2 baterías de 24 VDC/ 5 A
- 1 baterías de 24 VDC/ 8 A

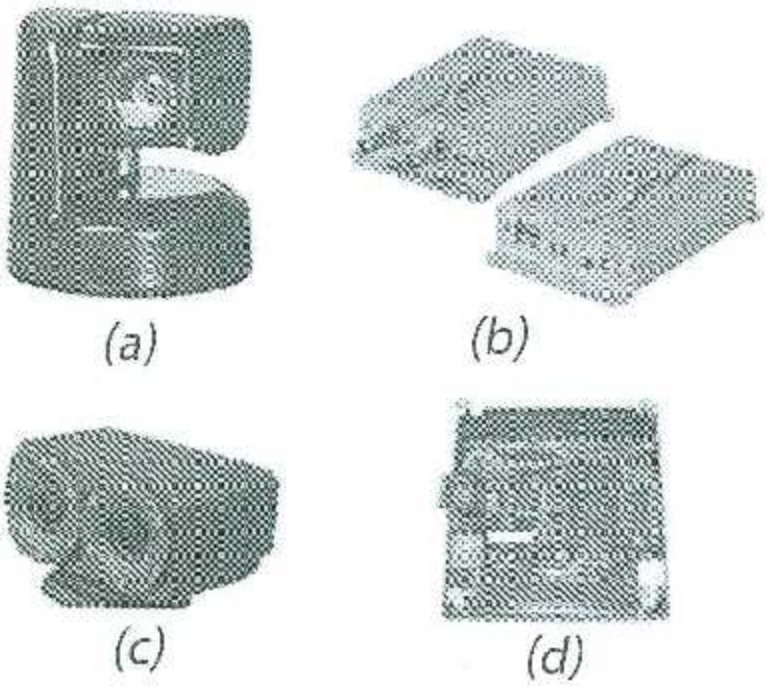

Figura 3.2: (a) Cámara PTZ marca RVision modelo SEE HP, (b) Framegrabber Sensoray 2553, (c) Cámara estéreo Mobile Ranger C3D, (d) Tarjeta Focus Robotics nDepth con interfaz PC104

Para recargar las baterías se cuenta con una fuente marca "Xantrex". Esta fuente, que se alimenta con 220 volts a 30 amperes, puede configurarse mediante un acomodo de jumpers que se encuentran en la parte trasera del equipo. La configuración actual se llama "modo de operacion remota" permitiendo que el microprocesador del robot controle la recarga de baterías. Para que la fuente pueda comenzar con la recarga de las baterías es primordial que el robot tenga al menos una carga de 20 volts para mandar las señales a la fuente. En caso de no contar con este nivel de energía el microprocesador no emitirá las señales de control para la fuente. La fuente debe de estar alimentada, encendida y conectada al robot antes de encender el robot.

3.1.1. Modos de operación

El robot presenta dos modos de operación, omnidireccional (véase Figura 3.3) y diferencial (véase Figura 3.4). En el modo de operación omnidireccional cada una de las cuatro llantas del robot trabaja de manera independiente, y puede tener una orientación diferente. El robot ya contiene un controlador omnidireccional de fábrica que orienta las ruedas de acuerdo a tres valores ingresados por el usuario que son: velocidad de avance o de traslación (v_x), velocidad lateral (v_y) y velocidad de rotación (ω). En este modo el robot puede moverse sin modificar su orientación.

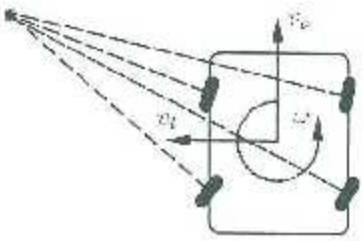

Figura 3.3: Modo de operación omnidireccional

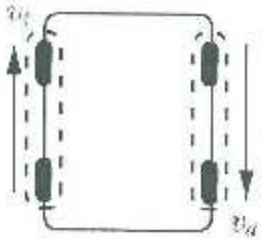

Figura 3.4: Modo de operación diferencial

En el modo de operación las ruedas se comportan como ruedas fijas, cuando regresan

a posición de *home*. Una vez en posición de *home* no pueden cambiar su orientación hasta cambiar al modo de operación omnidireccional. Los valores que se ingresan en este modo de operación es velocidad izquierda (v_i) y velocidad derecha (v_d).

3.2. Programación del robot (ARIA)

La carga de rutinas al robot se realiza mediante una herramienta para desarrollo de software (SDK en inglés) conocida como ARIA (por "Advanced Robot Interface for Application"). ARIA es básicamente una biblioteca de funciones escrita para los lenguajes de programación C/C++ y Python; para este trabajo se empleó el lenguaje C/C++ y el compilador "GCC" que viene preinstalado en la computadora a bordo. Esta biblioteca contiene funciones para indicar al robot que llegue a una determinada posición, que siga una ruta predefinida, que se mueva a una velocidad constante, o con una aceleración deseada, o que haga un mapa de la zona donde se encuentra detectando los obstáculos, por medio del laser que se encuentra en la parte frontal del robot. Todas las órdenes que se mandan al robot pueden hacerse de manera local si se carga el programa en la computadora a bordo, pero también existe la opción de crear una interfaz con la misma herramienta ARIA para enviar las órdenes de manera remota, empleando comunicación inalámbrica. La biblioteca aún se encuentra incompleta, ya que no tiene soporte para todas las cámaras, como la cámara monocular RVision que se encuentra montada en la parte superior del robot, por lo que se tiene que hacer uso de las bibliotecas como V4L2 y OpenCV para hacer la adquisición de imágenes y el tratamiento de las mismas.

Es posible tomar lecturas de los sensores como GPS, LRF (Laser Range Finder), encoders, giroscopio, velocidades, desplazamientos, orientación, etc. En la carpeta "\usr\local\Aria" se pueden encontrar los archivos de cabecera (.h), archivos de objetos compartidos (.so), la documentación y algunos ejemplos de programas que pueden ser modificados y compilados. Para obtener las lecturas del desplazamiento del robot, el microcontrolador realiza una estimación empleando una técnica de odometría obteniendo las lecturas de los encoders de cada una de las llantas y de un giroscopio de eje único para estimar la orientación

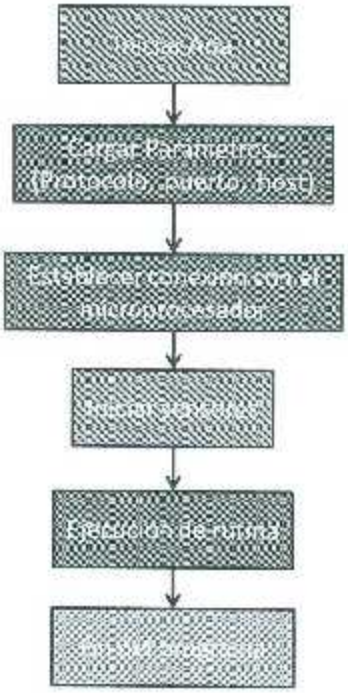

Figura 3.5: Pasos para una ejecución de una rutina en el robot Seekur

y el desplazamiento del robot. Para medir la velocidad se cuenta con un centro de acelerómetros, los cuales son utilizados no únicamente en las lecturas de las funciones ARIA, sino que también son utilizados por el controlador dinámico que tiene de fábrica el robot. El GPS es de marca Trimble, según especificaciones del fabricante, requiere de cuando menos tres comunicaciones satelitales para obtener la posición, presentando un error de aproximadamente dos metros. Para determinar si el GPS establece la comunicación, se puede observar en el dispositivo que se encuentra en la parte superior del robot un LED, que parpadea en color ámbar cuando no tiene conexión con almenos tres satélites, y cuando está en color verde es que tiene comunicación con tres o más satélites. El LRF es un láser de ocho milímetros de ancho y que se encuentra a una altura de 58 centímetros; tiene la capacidad de detectar objetos a una distancia de diez metros en un rango de 270°.

El programa que ejecuta todos los pasos de la Figura 3.5 se llama **main**, está escrito en C/C++. Al ejecutarse este programa primero se inicia el servicio de ARIA mediante el comando *Aria::init*. Luego para cargar parámetros del robot, ya sea de manera local

o remota, se debe de crear un objeto de la clase *ArArgumentParser* y posteriormente ejecutar la función miembro *loadDefaultArgument()*. Después para establecer la conexión con el microprocesador se debe de crear un objeto de la clase *ArRobotConector*, asociarlo al objeto con los parámetros y finalmente ejecutar la función *connectRobot()*.

Dentro del programa, el robot es un objeto de la clase *ArRobot*. Dado que la biblioteca ARIA utiliza la programación multihilo, el movimiento del robot inicia al ejecutarse el hilo asociado al objeto robot, mediante la función *runAsync(true)*. Algunas de las acciones que se le pueden indicar al robot son:

- Ingresar una posición y orientación. También es posible leer en cualquier instante la posición y orientación actual del robot.
- Medir tiempos en milisegundos y conocer cuánto ocupa cada ciclo de trabajo con la finalidad de conocer el diferencias de tiempo.
- Ingresar en el modo de operación omnidireccional, la velocidad de avance, la velocidad lateral y la velocidad de rotación. Las velocidades de avance y lateral se ingresan en mm/s, mientras que la velocidad de rotación se ingresa en grados por segundo.
- Tomar la lectura, en cualquier instante, de las velocidades antes mencionadas manejando las mismas unidades, velocidad lateral y velocidad de rotación.
- Seleccionar una aceleración deseada, ya que, según especificaciones del fabricante, se considera una aceleración constante; es posible cambiar este valor ingresándolo en mm/s^2.
- Para activar el modo de operación diferencial se ingresan dos velocidades; la velocidad de la rueda izquierda y la velocidad de la rueda derecha; también se ingresan estos valores en mm/s.
- Tomar lectura de la cantidad de desplazamiento que ha tenido el robot desde el momento en que fue encendido, este dato se regresa en mm. Para esto se tienen dos opciones; la primera es con una técnica de odometría, donde haciendo uso de

los encoders y del giroscopio de eje único se hace una estimación de este dato; la segunda es a través del GPS a bordo y puede obtenerse, el desplazamiento así como la latitud, longitud y altitud actuales del robot.

- Controlar algunas cámaras, como la PTZ (pan tilt zoom) que se encuentra montada en la parte superior del robot. También se tiene soporte por otra clase de cámaras.
- Adquirir información sobre el ambiente en el que se encuentra Seekur por medio del sensor LRF. Este sensor indica la distancia y el ancho del objeto que se encuentre frente al robot.

Para correr un programa escrito en ARIA en un sistema operativo Linux, se debe guardar el archivo con extensión *.cpp*. Para compilar se realiza a través del paquete GCC, para esto hay que posicionarse en la dirección "\usr\local\Aria" y correr el comando *make dirección_del_archivo*, donde la *dirección_del_archivo* debe de ser la ruta completa y nombre del archivo con extensión *.cpp*. Después para correr un ejecutable, se emplea la instrucción "*dirección_del_archivo parámetros*", en esta instrucción *dirección_del_archivo* es la ruta del archivo omitiendo la extensión *.cpp*. Los parámetros que acepta ARIA se muestran en la tabla 3.1.

3.3. Adquisición y procesamiento de imagenes

Para realizar la adquisición de las imagenes se requiere de una biblioteca llamada V4L2. La conexión de la cámara no se realiza de manera directa, ya que ésta entrega una señal analógica, por lo que está conectada a una adquisitora de imágenes Sensoray 2253. Este dispositivo, con interfaz USB, se encarga de digitalizar las imágenes para que sean reconocidas por las computadora de a bordo. El programa para realizar la comunicación está escrito en el lenguaje de programación C/C++. Se utilizó el programa *capture* que se instala junto con los controladores de la adquisitora de imagenes y también contiene el codigo fuente. Este programa fue modificado, ya que captura el video y lo guarda en formato *AVI*, *MP4*, *MPEG*, o en serie de imagenes *JPEG* numerando cada imagen con un

Tabla 3.1: Parámetros para ejecutar una rutina de ARIA

Instrucción	valor
-remoteHost <IP o Nombre de servidor>(abreviado -rh)	Conecta con el servidor de manera remota, bien sea por LAN o por comunicación serial.
-robotPort <Puerto Serial>(abreviado -rp)	Se especifica el puerto serial a utilizar. El puerto de predeterminado es el COM1
-robotBaud <velocidad en baudios por segundo>(abreviado -rb)	Se especifica la velocidad del puerto serial. El valor predeterminado es 9600.
-remoteRobotTcpPort <Puerto>(abreviado -rrtp)	Se especifica el puerto para conexión remota. El puerto predeterminado es 8101.

prefijo especificado previamente. Se modificó el programa para que fuera guardando cada imagen en extensión *JPG* guardándose siempre con el nombre de *foto.jpg*, reescribiendo la información ya existente.

El procesamiento de imágenes se realizó con OpenCV. La rutina con OpenCV se programó de acuerdo al diagrama de la figura 3.6. El bloque con la leyenda *Lectura del archivo 'foto.jpg'* se traduce al comando *image = iamread("foto.jpg", CV_LOAD_IMAGE_COLOR)* que carga la imagen a la variable *image*. Para especificar el umbral de color se va a utilizar la representación HSV (hue, saturation and value, que en español sigifica matiz, saturación y valor) ya que es más sencillo seleccionar un color en ese formato que en RGB, aunque OpenCV lo termina transformando en valores RGB. La segmentación se realiza con el comando

```
cvInRangeS(image, cvScalar(lowerH,lowerS,lowerV), cvScalar(upperH,upperS,upperV),
    imgThresh)
```

donde *image* es la imagen original; (*lowerH*, *lowerS*, *lowerV*) son los valores para la cota inferior del umbral de color; (*upperH*, *upperS*, *upperV*) es la cota superior del umbral de color; e *imgThresh* es donde se almacena la imagen resultante. Los valores de *lowerH* y

upperH van de 0 a 180, y los valores de *lowerS*, *upperS*, *lowerV* y *upperV* van de 0 a 256.

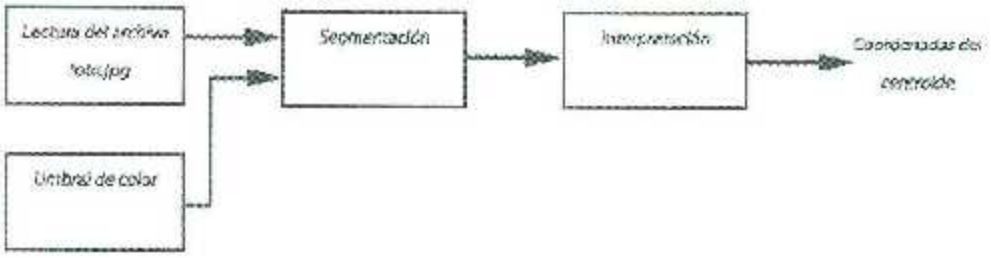

Figura 3.6: Secuencia del procesamiento de imagenes

Para la etapa de interpretación se calculan los momentos de la imagen que se presentan en la sección 2.2.2. Estos se obtienen con el comando *cvMoments(imgThresh, moments, 1)*, siendo la variable *moments* del tipo *CvMoments*[8]. Las coordenadas del centroide se obtienen con la ecuación (2.11) que se programa con el siguiente algoritmo

```
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
double area = cvGetCentralMoment(moments, 0, 0);

int posX = moment10/area;
int posY = moment01/area;
```

donde *posX*, *posY* son las coordenadas x, y del centroide.

Capítulo 4

Modelado cinemático

Para determinar el modelo cinemático de postura es necesario establecer ciertas condiciones.

a) El robot no contiene partes flexibles.

b) Las ruedas no se deforman y se mueven en un plano horizontal.

4.1. Modo de operación omnidireccional

Para este trabajo se obtuvo un modelo cinemático partiendo de las ecuaciones de cinemática para cuerpos rígidos de [Beer et al., 2007].

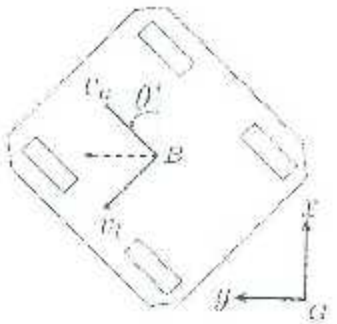

Figura 4.1: Diagrama de cuerpo libre del robot

Las velocidades que existen en el robot se muestran en las Figuras 4.1 y 4.2. Las señales de control son tres; la velocidad de avance frontal v_a; la velocidad lateral v_l y la velocidad de rotación ω, las tres son medidas respecto al marco B (marco del robot).

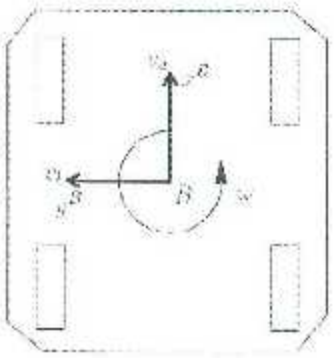

Figura 4.2: Velocidades de entrada del Robot Seekur

Tal como se explico en la Sección 2.1, para poder medir las velocidades desde el marco del mundo, es necesario tomar en cuenta la siguiente matriz de rotación

$$R_B^G = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

donde θ es el ángulo que indica la orientación del carro con respecto al marco del mundo. El vector de velocidades lineales del robot (véase Figura 4.2) se define como

$$\boldsymbol{v}^B = [\dot{x}^B \quad \dot{y}^B \quad \dot{z}^B]^T = [v_a \quad v_l \quad 0]^T. \tag{4.2}$$

Es posible obtener el vector de velocidades lineales visto desde el marco G

$$\boldsymbol{v}^G = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_B^G v^B = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_a \\ v_l \\ 0 \end{bmatrix} = \begin{bmatrix} v_a\cos\theta - v_l\sin\theta \\ v_a\sin\theta + v_l\cos\theta \\ 0 \end{bmatrix} \tag{4.3}$$

donde $\dot{x}$, $\dot{y}$, $\dot{z}$ son las velocidades lineales medidas desde el marco G. El vector de velocidades angulares del carro es

$$\omega^B = [\omega_x^B \quad \omega_y^B \quad \omega_z^B]^T = [0 \quad 0 \quad \dot{\theta}]^T. \tag{4.4}$$

Para obtener el vector de velocidades angulares visto desde el marco G se realiza la siguiente multiplicación

$$\omega^G = R_B^G \omega^B = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}, \tag{4.5}$$

es decir que el vector de velocidades angulares visto desde el marco G es igual al visto desde el marco B. Conociendo las velocidades lineales y tomando en cuenta que $\dot{\theta} = \omega$, el modelo cinemático de postura se escribe de la siguiente forma

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_a \\ v_l \\ \omega \end{bmatrix}. \tag{4.6}$$

De la ecuación (4.6) es posible reconocer el jacobiano como viene descrito en la [Siciliano et al., 2009] donde el modelo cinemático debe de estar escrito en la forma

$$\dot{\boldsymbol{p}} = J\dot{\boldsymbol{q}}, \tag{4.7}$$

donde $\dot{\boldsymbol{p}}$ es el vector de señales de salida, $dot\boldsymbol{q}$ es el vector de señales de entrada y J es la matriz jacobiana. Por lo tanto la matriz jacobiana para el robot está descrita como

$$J = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Para conocer las velocidades lineales a las cuales se mueven cada una de las ruedas v_1, v_2, v_3 y v_4 a partir de las velocidades v_a, v_l y ω. La distribución de las ruedas se muestra en la Figura 4.3 y se tienen las siguientes expresiones:

$$v_1 = \begin{bmatrix} v_{1x} \\ v_{1y} \end{bmatrix} = \begin{bmatrix} v_a \\ v_l \end{bmatrix} + \omega \begin{bmatrix} -\frac{b}{2} \\ \frac{a+b}{2} \end{bmatrix} \tag{4.9}$$

$$v_2 = \begin{bmatrix} v_{2x} \\ v_{2y} \end{bmatrix} = \begin{bmatrix} v_a \\ v_l \end{bmatrix} + \omega \begin{bmatrix} \frac{b}{2} \\ \frac{a+b}{2} \end{bmatrix} \tag{4.10}$$

$$v_3 = \begin{bmatrix} v_{3x} \\ v_{3y} \end{bmatrix} = \begin{bmatrix} v_a \\ v_l \end{bmatrix} + \omega \begin{bmatrix} -\frac{b}{2} \\ -\frac{a+b}{2} \end{bmatrix} \tag{4.11}$$

$$v_4 = \begin{bmatrix} v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} v_a \\ v_l \end{bmatrix} + \omega \begin{bmatrix} \frac{b}{2} \\ -\frac{a+b}{2} \end{bmatrix}, \tag{4.12}$$

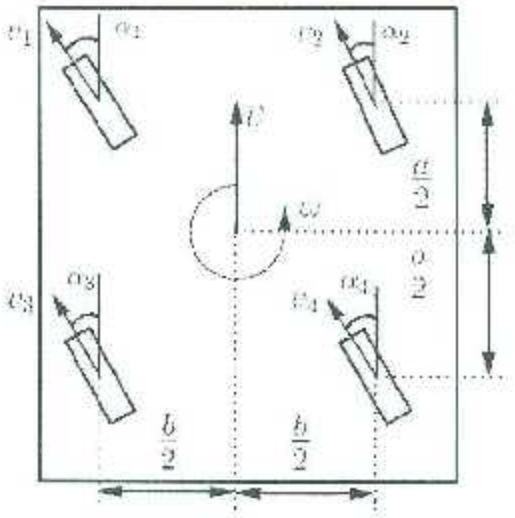

Figura 4.3: Distribución de las ruedas en el robot Seekur.

donde las variables v_{1x}, v_{1y}, v_{2x}, v_{2y}, v_{3x}, v_{3y}, v_{4x} y v_{4y} corresponden a las componentes en x e y de la i-ésima rueda en el marco del robot. Para conocer la orientación de cada una de las ruedas, ésta se puede calcular con la función $\text{atan2}(v_{iy}, v_{ix})$. Estos ángulos son llamados α_1, α_2, α_3 y α_4 como se muestra en la Figura 4.3.

El movimiento de un robot, de forma alternativa, puede ser representado por un centro de rotación instantáneo ó ICR (ver Figura 4.4), y se encuentra a una distancia c, que es el radio de la circunferencia que traza el robot al momento de tener un movimiento de rotación. Para determinar el radio, se utiliza la expresión $c = \frac{v}{\omega}$ donde v es la velocidad lineal del carro y ω es la velocidad de rotación del carro, por lo cual se puede determinar que si únicamente se tiene el movimiento de traslación $[v \neq 0 \quad \omega = 0]$ el radio del ICR se encuentra a una distancia infinita y, en cambio, si únicamente se tiene un movimiento de rotación sobre su propio eje $[v = 0 \quad \omega \neq 0]$, el radio al que se encuentra el ICR es cero.

Para conocer los ángulos que deben de tener las llantas para realizar un movimiento considerando el ICR se emplean las siguientes ecuaciones

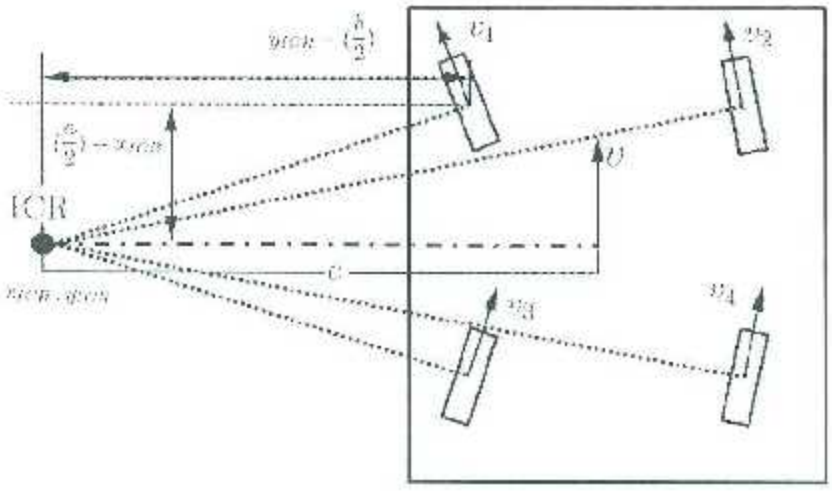

Figura 4.4: Velocidades de las ruedas.

$$\alpha_1 = \text{atan2}\left(\left(\frac{a}{2}\right) - x_{ICR}, y_{ICR} - \left(\frac{b}{2}\right)\right) \tag{4.13}$$

$$\alpha_2 = \text{atan2}\left(\left(\frac{a}{2}\right) - x_{ICR}, y_{ICR} + \left(\frac{b}{2}\right)\right) \tag{4.14}$$

$$\alpha_3 = \text{atan2}\left(-\left(\frac{a}{2}\right) - x_{ICR}, y_{ICR} - \left(\frac{b}{2}\right)\right) \tag{4.15}$$

$$\alpha_4 = \text{atan2}\left(-\left(\frac{a}{2}\right) - x_{ICR}, y_{ICR} - \left(\frac{b}{2}\right)\right) \tag{4.16}$$

en donde x_{ICR} y y_{ICR} son las coordenadas donde se encuentra el ICR visto desde el marco B. Para calcular la curvatura del movimiento se tiene $\gamma = \frac{1}{\sqrt{x_{ICR}^2 + y_{ICR}^2}}$. Cuando el robot está en movimiento se pueden cumplir algunos de los siguientes tres casos, los cuales indican si el robot va a tener un movimiento rectilíneo, curvilíneo o rotación.

1. Si $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$ significa que el robot tiene un movimiento de traslación (movimiento rectilíneo o movimiento curvilíneo) siendo $\omega = 0$ y $v \neq 0$

2. Si $\alpha_1 = -\arctan\left(\frac{a}{b}\right)$, $\alpha_2 = \arctan\left(\frac{a}{b}\right)$, $\alpha_3 = \arctan\left(\frac{a}{b}\right)$ y $\alpha_4 = -\arctan\left(\frac{a}{b}\right)$ el robot rota sobre su propio eje siendo $\omega \neq 0$ y $v = 0$.

3. Si la orientacion de las ruedas no cumple con ninguna de las dos condiciones anteriores entonces se tiene un movimiento de rotación con un ICR fuera del eje siendo

$\omega \neq 0$ y $v \neq 0$.

4.2. Modo de operación diferencial

Para este modo de operación es posible asignarle al par de ruedas de la derecha una misma velocidad y al par de ruedas de la izquierda otra velocidad. Las posición del robot se especifica en coordenadas cartesianas $[x \quad y]$, el ángulo de dirección se designa como θ, b es la separación entre las ruedas derechas e izquierdas, v la velocidad lineal, ω velocidad angular, v_i velocidad lineal de las ruedas izquierdas, ω_i velocidad angular de las ruedas izquierdas, v_d velocidad lineal de las ruedas derechas, ω_d velocidad de giro de las ruedas derechas, r radio de la rueda, c distancia del ICR al centro del robot y $r = \frac{1}{\text{ICR}} = \frac{1}{c}$ curvatura (véase Figura 4.5).

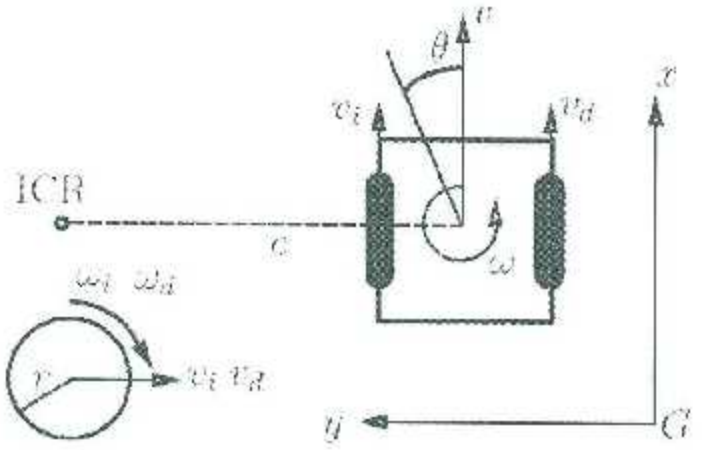

Figura 4.5: Esquema de un robot móvil con tracción diferencial

El análisis siguiente supone que cada par de ruedas (derecha e izquierda) del robot se puede ver como una sola rueda de manera que se cumpla la codición de no deslizamiento. En realidad, sin embargo, si existe deslizamiento de las ruedas en este modo de operación. Al momento de que un carro con tracción diferencial realiza una curva, el punto más

alejado tiene una velocidad lineal más alta, que un punto más cercano al centro de la circunferencia, como se muestra en la Figura 4.6.

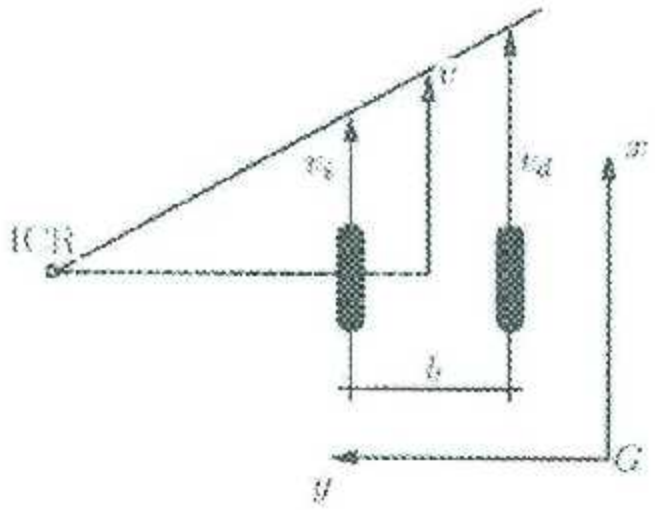

Figura 4.6: Velocidades lineales respecto al ICR

Por lo tanto, para obtener la velocidad lineal de cada una de las ruedas, a partir de una velocidad de carro deseada y una curvatura deseada se utilizan las siguientes ecuaciones [Ollero, 2007]:

$$v_d = \omega r \left(c + \frac{b}{2}\right) = \frac{v}{c}\left(c + \frac{b}{2}\right) \tag{4.17}$$

$$v_i = \omega r \left(c - \frac{b}{2}\right) = \frac{v}{c}\left(c - \frac{b}{2}\right). \tag{4.18}$$

En caso de requerir la velocidad lineal en función de las ruedas se utiliza la siguiente igualdad:

$$v = \frac{v_d - v_i}{2}. \tag{4.19}$$

La velocidad ω se calcula con

$$\omega = \frac{v_d - v_i}{b}. \tag{4.20}$$

Las velocidades para el giro de las ruedas se definen como

$$\omega_d = \frac{v_d}{r} - \frac{v\gamma}{r}\left(c + \frac{b}{2}\right) \tag{4.21}$$

$$\omega_i = \frac{v_i}{r} - \frac{v\gamma}{r}\left(c - \frac{b}{2}\right). \tag{4.22}$$

Por otra parte, hay que recordar que el modelo cinemático del robot móvil tipo diferencial es

$$\dot{x} = v\cos\theta \tag{4.23}$$

$$\dot{x} = v\sin\theta \tag{4.24}$$

$$\dot{\theta} = \omega \tag{4.25}$$

De modo que usando (4.19) y (4.20) se llega a

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} \\ \frac{1}{b} \end{bmatrix} v_d - \begin{bmatrix} \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} \\ -\frac{1}{b} \end{bmatrix} v_i, \tag{4.26}$$

o en forma matricial

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{r\sin\theta}{2} & \frac{r\sin\theta}{2} \\ \frac{r}{b} & -\frac{r}{b} \end{bmatrix} \begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ \frac{r}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} v_d \\ v_i \end{bmatrix}. \tag{4.27}$$

Por lo tanto el jacobiano, las señales de entrada y las señales de salida están dadas respectivamente como

$$J(p) = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ \frac{r}{b} & -\frac{r}{b} \end{bmatrix} \quad \dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad \dot{q} = \begin{bmatrix} v_d \\ v_i \end{bmatrix}. \tag{4.28}$$

4.3. Cinemática de la cámara

En la Figura 4.7 se muestra la distribución de los marcos asignados a la cámara RVision PTZ. El marco 0 se asignó con la misma orientación que el marco del robot, colocándolo en la base de la cámara. El marco 3, que es el marco de cámara en sí, esta asignado de

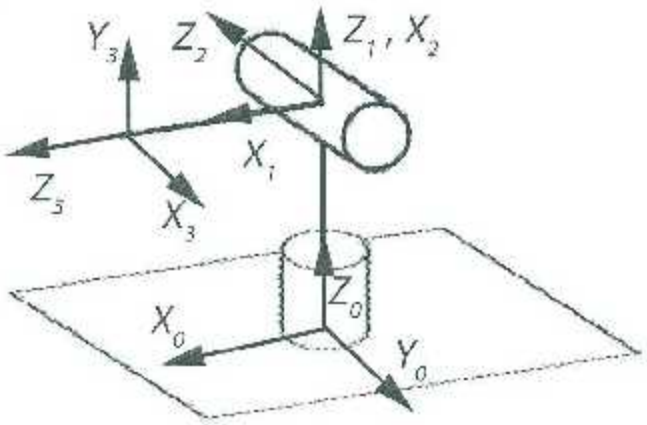

Figura 4.7: Distribución de marcos de la cámara

acuerdo al marco utilizado para obtener el jacobiano de la imagen obtenido en el capítulo 2. Para asignar los marcos 1 y 2, y encontrar las matrices de transformación se emplea la convención Denavit-Hartenberg modificada por Khalil ([Khalil and Kleinfinger, 1986]).

Tabla 4.1: Parámetros Denavit-Hartenberg de la cámara

i	α_i	d_i	θ_i	r_i
1	0	0	θ_1	d_1
2	90	0	$\theta_2 - 90$	0
3	90	0	-90	d_3

Los parámetros cinemáticos obtenidos se muestran en la tabla 4.1, y con ellos se pueden calcular las matrices de rotación relativas entre cada par de marcos consecutivos. Las matrices de rotación son

$$R_1^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_2^1 = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \quad R_3^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \tag{4.29}$$

Para obtener la matriz de rotación total se realiza la siguiente multiplicación

$$R_3^0 = R_1^0 R_3^1 R_3^2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.30}$$

Tomando en cuenta que el marco 0 se asignó igual que el marco B la matriz de rotación entre estos dos marcos es igual a la matriz identidad ($R_0^B = \mathrm{I}$). Para conocer la rotación de la cámara respero al marco del mundo se realiza la siguiente operación

$$R_3^G = R_B^G R_0^B R_3^0 = \begin{bmatrix} -\sin\theta & 0 & \cos\theta \\ \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \end{bmatrix}, \tag{4.31}$$

Para convertir las velocidad lineal del robot en el marco de la cámara se emplea la ecuación

$$v^c = R_G^3 v^g \tag{4.32}$$

donde v^c es el vector de velocidad lineal visto en el marco de la cámara (marco 3) de traslación vistas desde el marco 3 y se cumple la propiedad $R_G^3 = (R_3^G)^{-1} = (R_3^G)^T$, de modo que

$$R_G^3 = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \\ \cos\theta & \sin\theta & 0 \end{bmatrix}, \tag{4.33}$$

El vector de velocidad lineal visto desde el marco de la cámara entonces resulta ser

$$v^c = \begin{bmatrix} \dot{x}^c \\ \dot{y}^c \\ \dot{z}^c \end{bmatrix} - R_G^3 \begin{bmatrix} \dot{x} \\ y \\ z \end{bmatrix} = \begin{bmatrix} y\cos\theta - \dot{x}\sin\theta \\ 0 \\ \dot{x}\cos\theta + \dot{y}\sin\theta \end{bmatrix}. \tag{4.34}$$

Para obtener el vector de velocidad angular visto desde el marco de la cámara se emplea la expresion

$$\omega^c = \begin{bmatrix} \omega_x^c \\ \omega_y^c \\ \omega_z^c \end{bmatrix} = R_G^3 \omega^G = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}, \tag{4.35}$$

donde ω^G es dada en 4.5. Para conocer el movimiento de los puntos proyectados en e. sensor que está dentro de la cámara se aplica la siguiente ecuación [Siciliano et al., 2009]

$$\dot{s} = \begin{bmatrix} \dot{x}^s \\ \dot{y}^s \end{bmatrix} = L \begin{bmatrix} v^c \\ \omega^c \end{bmatrix}, \tag{4.36}$$

donde $\dot{s}$ son las velocidades de los puntos en [m/s] y L es el jacobiano escrito en (2.24). Realizando los productos se obtiene

$$\dot{s} = \begin{bmatrix} \frac{f}{z^c} & 0 & \frac{x^c}{z^c} & \frac{x^c y^c}{f} & -\frac{f^2+(x^c)^2}{f} & y^c \\ 0 & -\frac{f}{z^c} & \frac{y^c}{z^c} & \frac{f^2+y^c}{f} & -\frac{x^c y^c}{f} & -x^c \end{bmatrix} \begin{bmatrix} \dot{x}^c \\ \dot{y}^c \\ \dot{z}^c \\ \omega_x^c \\ \omega_y^c \\ \omega_z^c \end{bmatrix} \tag{4.37}$$

$$\dot{x}^s = \cos\theta\left(\frac{x^c\dot{x}^c - f\dot{y}^c}{z^c}\right) - \sin\theta\left(\frac{x^c y^c + f\dot{x}^c}{z^c}\right) - \frac{\dot{\theta}^c(f^2+(x^s))}{f} \tag{4.38}$$

$$\dot{y}^s = \frac{y^c}{z^c}(\dot{x}^c\cos\theta - \dot{y}^c\sin\theta) - \dot{\theta}^c\frac{x^c y^c}{f}. \tag{4.39}$$

Las unidades del vector de salida están dadas en metros pero las unidades que utiliza una imagen son [pixeles], así que se requiere de una conversion. Un pixel no es una medida definida en el mundo real (una misma imagen puede cambiar su resolución en pixeles y seguir representando las mismas dimensiones en el mundo real). Como se está tomando un modelo lineal y también se conoce la resolución de la cámara RVision PTZ(Pan Tilt Zoom) que es de 320 x 480 pixeles, para conocer el factor de conversión del sensor de la cámara, se emplea un patrón de cuadros como el mostrado en la figura 4.8 y seguir el procedimiento descrito en [Siciliano et al., 2009].

La relación entre las coordenadas del mundo real (x_s, y_s) y las coordenadas en el plano de la imagen (x_p, y_p) están dadas por

$$x_p = c_x x_s \tag{4.40}$$

$$y_p = c_y y_s. \tag{4.41}$$

Tras seguir el procedimiento descrito en [Siciliano et al., 2009] se obtuvo $c_x = 21.886$ pixeles/metro y $c_y = 21.111$ pixeles/metro. Así que si se seleccionan como variables de estado las siguientes

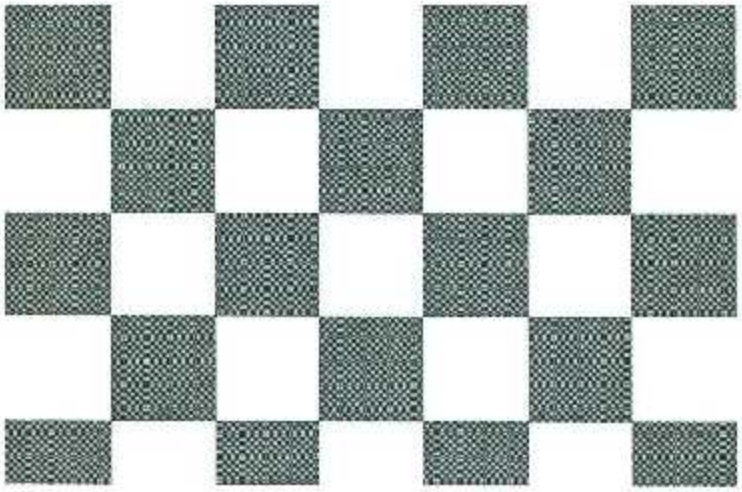

Figura 4.8: Patrón empleado para la calibracion de cámaras

- Variables de estado

$$\begin{aligned} x_1 &= c_x x_s \\ x_2 &= -c_y y_s \\ x_3 &= \theta. \end{aligned} \tag{4.42}$$

- Variables de entrada

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2 \\ \dot{\theta} &= u_3. \end{aligned} \tag{4.43}$$

entonces las ecuaciones de estado quedan

$$\dot{x}_1 = \left(\cos x_3\left(\frac{x_1 u_1 - c_x f u_2}{z^c}\right) - \sin x_3\left(\frac{x_1 u_2 + c_x f u_1}{z^c}\right)\right) + c_x u_3 f + u_2 \frac{(x_1)^2}{c_x f} \tag{4.44}$$

$$\dot{x}_2 = -\frac{x_2}{z^c}\left(u_1 \cos x_3 + u_2 \sin x_3\right) - u_3 \frac{x_1 x_2}{c_x f} \tag{4.45}$$

$$\dot{x}_3 = u_3. \tag{4.46}$$

Las expresiones anteriores son el modelo en espacio de estados donde el estado x_1 y x_2 son las posiciones x y y del centro del objeto a seguir, medidas en el espacio de la

imager, $\dot{x}_1$ y $\dot{x}_2$ son las velocidades del centro del objeto medidas en *pixeles/segundo*. x_3 es la orientación del robot visto desde el marco G medido en *grados* y $\dot{x}_3$ es la velocidad angular del robot medida en *grados/segundo*.

Capítulo 5

Control

En esta capítulo se describe el controlador usado para implementar los algoritmos de control servo-visual en el robot Seekur. Se emplea un esquema de doble lazo como el mostrado en la Figura 5.1. El control cinemático recibe como entrada las características deseadas de la imagen y las características actuales de la imagen, teniendo como salida la velocidad lineal y angular del robot. Por otra parte el control dinámico recibe como entrada el error entre la velocidad lineal y angular deseadas y, la velocidad lineal y angular reales del robot, entregando las fuerzas y torques necesarios para el movimiento del robot.

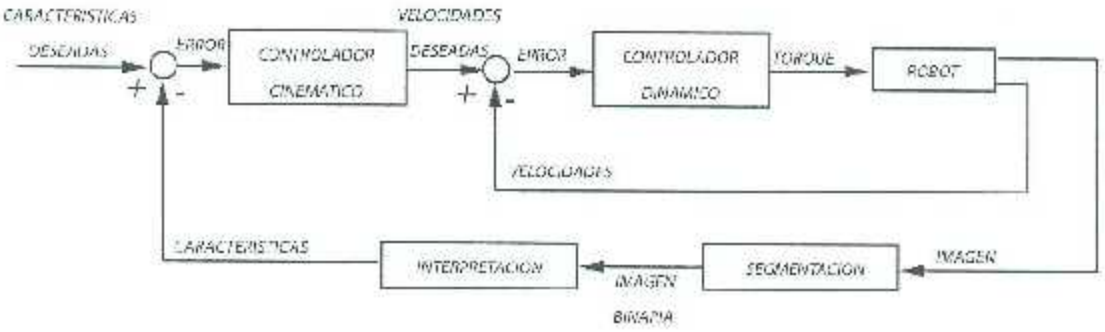

Figura 5.1: Diagrama a bloques

5.1. Control cinemático

La ley de control no lineal para el control cinemático (lazo externo) a utilizar es de tipo PD y emplea exponenciales como saturación. En esta ley de control tanto la acción proporcional como la derivativa son acotadas, por una constante $P_M \in \mathbb{R}_+$ para la acción proporcional y por una constante $D_M \in \mathbb{R}_+$ para la acción derivativa. La expresión a utilizar es la siguiente:

$$u_2 = P_M \text{Sign}(\tilde{x})(1 - e^{-k_p|\tilde{x}|}) - P_D \text{Sign}(\dot{\tilde{x}})(1 - e^{-k_d|\dot{\tilde{x}}|}) \tag{5.1}$$

donde se usa la función Sign(x) definida como

$$\text{Sign}(x) = \begin{cases} 1 & \forall \quad x \geq 0, \\ -1 & \forall \quad x < 0 \end{cases}, \tag{5.2}$$

$\tilde{x} = x_{2d} - x_2$ es el error proporcional en pixeles, $\dot{\tilde{x}}$ es el error derivativo en $\frac{\text{pixeles}}{\text{segundos}}$, $k_p \subset \mathbb{R}$ es la ganancia proporcional en $\frac{\text{metros}}{\text{pixel segundos}}$ y $k_d \in \Re_+$ es la ganancia derivativa en $\frac{\text{metros}}{\text{pixeles}}$.

Para demostrar que la ley de control es diferenciable se hace el siguiente análisis.

- Obsérvese que la acción proporcional

$$u_p = P_M \text{Sign}(\tilde{x})(1 - e^{-k_p|\tilde{x}|}) \tag{5.3}$$

puede reescribir como

$$u_p = \begin{cases} P_M(1 - e^{-k_p \tilde{x}}) & \forall \quad \tilde{x} \geq 0, \\ -P_M(1 - e^{k_p \tilde{x}}) & \forall \quad \tilde{x} < 0 \end{cases}.$$

Tomando la derivada de cada segmento

$$\frac{du_p}{dt} = \begin{cases} P_M k_p e^{-k_p \tilde{x}} & \forall \quad \tilde{x} \geq 0, \\ P_M k_p e^{k_p \tilde{x}} & \forall \quad \tilde{x} < 0 \end{cases},$$

y se observa que al tomar el límite $\tilde{x} \to 0$ la derivada por ambos lados coincide, de manera que u_p es continua y diferenciable cuando menos una vez.

Para la acción derivativa se realiza el mismo análisis.

- Acción derivativa

$$u_d = P_D \text{Sign}(\dot{\tilde{x}})(1 - e^{-k_d|\dot{\tilde{x}}|}) \tag{5.4}$$

que se puede reescribir como

$$u_d = \begin{cases} P_D(1 - e^{-k_d\dot{\tilde{x}}}) & \forall \quad \dot{\tilde{x}} > 0, \\ -P_D(1 - e^{k_d\dot{\tilde{x}}}) & \forall \quad \dot{\tilde{x}} < 0 \end{cases},$$

Tomando la derivada para cada segmento:

$$\frac{du_d}{dt} = \begin{cases} P_D k_d e^{-k_d\dot{\tilde{x}}} & \forall \quad \dot{\tilde{x}} \geq 0, \\ P_D k_d e^{k_d\dot{\tilde{x}}} & \forall \quad \dot{\tilde{x}} < 0 \end{cases},$$

tomando el límite de $\dot{\tilde{x}} \to 0$ la derivada por ambos lados coincide. Al igual que la parte proporcional se demuestra que es diferenciable cuando menos una vez.

5.2. Controlador dinámico

Como controlador dinámico, se supone, para fines de simulación, el siguiente:

$$u_a = k_{p1}(e_a) - k_{d1}(\dot{e}_a) + k_{i1}\int e_a \tag{5.5}$$

$$u_l = k_{p2}(e_l) - k_{d2}(\dot{e}_l) + k_{i2}\int e_l \tag{5.6}$$

$$u_r = k_{p3}(e_\omega) - k_{d3}(\dot{e}_\omega) + k_{i3}\int e_\omega \tag{5.7}$$

donde

$$e_a = v_{ad} - v_a \tag{5.8}$$

$$e_l = v_{ld} - v_l \tag{5.9}$$

$$e_\omega = \omega_d - \omega_r \tag{5.10}$$

donde v_{ad}, v_{ld} y ω_d son respectivamente las velocidades de avance deseada, la velocidad lateral deseada y la velocidad angular deseada; v_a, v_l y ω son la velocidad de avance real, la velocidad lateral real y la velocidad angular real; k_{pj}, k_{dj} y k_{ij} (con $j = 1, 2, 3$) son las ganancias proporcional, derivativa e integral de cada componente de velocidad; u_a, u_l

y u_r son las salidas del controlador, corresponderan a la fuerza aplicada en dirección del movimiento de avance, la fuerza aplicada en la dirección del movimiento lateral y el torque aplicado sobre el eje de rotación del robot.

Para la simulación, sin embargo, no se utilizo el controlador propiamente dado por 5.5, si no que se decidio emplear una aproximación lineal para el modelo dinámico del robot y encontrar una función de transferencia que incluya tanto el robot como el controlador dinámico. Esto se explica a continuación.

El modelo dinámico utilizado similar al sugerido en [Ollero, 2007] para el robot RAM-1 que es muy parecido al robo Seekur. Se consideran las siguientes ecuaciones

$$\dot{v}_a = -\frac{v_a}{\tau_a} + \frac{K u_a}{\tau_a} \tag{5.11}$$

$$\dot{v}_l = -\frac{v_l}{\tau_a} - \frac{K u_l}{\tau_a} \tag{5.12}$$

$$\dot{\omega} = -\frac{\omega}{\tau_\omega} - \frac{K u_r}{\tau_\omega}. \tag{5.13}$$

donde 5.11 es la dinámica para la velocidad de avance (v_a), la expresión 5.12 es la dinámica para la velocidad lateral (v_l) y la expresión 5.13 es la dinámica para la velocidad angular (ω), τ_a es una constante de tiempo para la velocidad lineal, τ_r es la constante de tiempo para la velocidad angular. K es una constante arbitraria. Reescribiendo el modelo en forma matricial la expresión es

$$\begin{bmatrix} \dot{v}_a \\ \dot{v}_l \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_a} & 0 & 0 \\ 0 & \frac{1}{\tau_a} & 0 \\ 0 & 0 & \frac{1}{\tau_\omega} \end{bmatrix} \begin{bmatrix} v_a \\ v_l \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{K}{\tau_a} & 0 & 0 \\ 0 & \frac{K}{\tau_a} & 0 \\ 0 & 0 & \frac{K}{\tau} \end{bmatrix} \begin{bmatrix} u_a \\ u_l \\ u_r \end{bmatrix}. \tag{5.14}$$

Sustituyendo la expresión (5.5) en (5.14)

$$\begin{bmatrix} \dot{v}_a \\ \dot{v}_l \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_a} & 0 & 0 \\ 0 & \frac{1}{\tau_a} & 0 \\ 0 & 0 & \frac{1}{\tau_\omega} \end{bmatrix} \begin{bmatrix} v_a \\ v_l \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{K}{\tau_a} & 0 & 0 \\ 0 & \frac{K}{\tau_a} & 0 \\ 0 & 0 & \frac{K}{\tau} \end{bmatrix} \begin{bmatrix} k_{p1} e_a + k_{d1} \dot{e}_a - k_{i1} \int e_i \\ k_{p2} e_l + k_{d2} \dot{e}_l + k_{i2} \int e_l \\ k_{p3} e_\omega - k_{d3} \dot{e}_\omega - k_{i3} \int e_\omega \end{bmatrix} \tag{5.15}$$

$$e_a = v_{ad} - v_a \qquad \dot{e}_a = \dot{v}_{ad} - \dot{v}_a = 0 - \dot{v}_a \tag{5.16}$$

$$e_l = v_{ld} - v_l \qquad \dot{e}_l = \dot{v}_{ld} - \dot{v}_a = 0 - \dot{v}_l \tag{5.17}$$

$$e_\omega = v_{\omega d} - v_\omega \qquad \dot{e}_\omega = \dot{\omega}_d - \dot{v}_a = 0 - v_\omega.$$

Aplicando la transformada de Laplace se tiene

$$\begin{bmatrix} v_a(s)s \\ v_i(s)s \\ \omega(s)s \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_v} & 0 & 0 \\ 0 & \frac{1}{\tau_v} & 0 \\ 0 & 0 & \frac{1}{\tau} \end{bmatrix} \begin{bmatrix} v_a(s) \\ v_i(s) \\ \omega(s) \end{bmatrix} + \begin{bmatrix} \frac{K}{\tau_v} & 0 & 0 \\ 0 & \frac{K}{\tau_v} & 0 \\ 0 & 0 & \frac{K}{\tau} \end{bmatrix} \begin{bmatrix} k_{p1}e_a(s) - k_{d1}e_a(s)s + k_{i1}\frac{e_a(s)}{s} \\ k_{p2}e_i(s) + k_{d2}e_i(s)s + k_{i2}\frac{e_i(s)}{s} \\ k_{p3}e_\omega(s) - k_{d3}e_\omega(s)s + k_{i3}\frac{e_\omega(s)}{s} \end{bmatrix}.$$

Desarrollando, se puede obtener la función de transferencia en lazo cerrado para cada lazo interno de la Figura 5.1

$$\begin{aligned} \frac{v_a(s)}{v_{ad}(s)} &= \frac{Kk_{d1}s - Kk_{i1}}{(\tau_v + Kk_{d1})s^2 + (Kk_{p1} + 1)s + Kk_{i1}} \\ \frac{v_i(s)}{v_{id}(s)} &= \frac{Kk_{p2}s - Kk_{i2}}{(\tau_v + Kk_{d2})s^2 + (Kk_{p2} + 1)s + Kk_{i2}} \\ \frac{\omega(s)}{\omega_d(s)} &= \frac{Kk_{p3}s + Kk_{i3}}{(\tau - Kk_{d3})s^2 + (Kk_{p3} - 1)s + Kk_{i3}}. \end{aligned} \tag{5.18}$$

Capítulo 6

Simulaciones y experimentos

En este capítulo se describen las simulaciones y experimentos realizadas con el robot Seekur. En ambos casos se considero una tarea muy simple, consiste en alinear las cuatro ruedas del robot de manera que realize un movimiento lateral y suponer que el objeto a seguir se mueve en el plano frente al robot. En otras palabras se supone que durante toda la tarea solo hay movimiento en el eje y y que unicamente se aplica como acción de control a la velocidad lateral (v_l) del robot.

6.1. Simulaciones

6.1.1. Regulación

Las simulaciónes se realizaron en Simulink empleando para el robot el modelo lineal descrito en la Sección 5.2. La primera simulación es para regulación, aplicándose varios escalones como referencias. La señal de referencia indica en que posición se encuentra el objeto respecto al marco del mundo. El objeto comienza con una posición de 0.3 (metros) cambiando en los segundos 55, 80, 100, 125 y 160 a una posición en metros de 0.6, 0.35, 0.7, 0.5 y 0.75 respectivamente. Se supone que el objeto a seguir se sitúa a una distancia z_o de 5 metros. Los resultados de esta simulación se muestran en las Figuras 6.1, 6.2 y 6.3.Los valores que se asignaron a los parámetros del controlador dinámico se muestran

en la Tabla 6.1.

Tabla 6.1: Ganancias sintonizadas

Ganancia / j	k_{pj}	k_{dj}	k_{ij}	K	τ
1	200	10	2	1	10
2	200	10	2	1	10
3	150	50	2	1	5

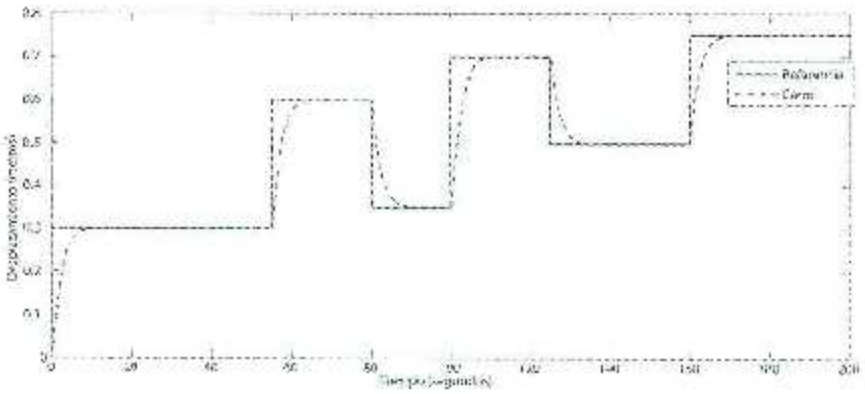

Figura 6.1: Gráficas obtenidas en la simulación de regulación

La gráfica que se observa en la Figura 6.1 corresponde al movimiento lateral del objeto y del carro. Se observa que el carro alcanza su objetivo en aproximadamente 10 segundos. En la Figura 6.2 se presenta la gráfica del error con escala en pixeles. El desplazamiento del objeto genera un error entre 50 y 30 pixeles. La señal de control se presenta en la Figura 6.3. Cada vez que se mueve el objeto la velocidad máxima calculada por el control es de 0.1 m/s y una vez alcanzado el objetivo la velocidad es cero. Los valores de las ganancias para regulación se muestran en la tabla 6.2.

6.1.2. Seguimiento

Para la simulación de seguimiento se escogió una onda cosenoidal como referencia. La onda cosenoidal es de 1 metro de amplitud y una frecuencia de 0.2 rad/seg. El tiempo de

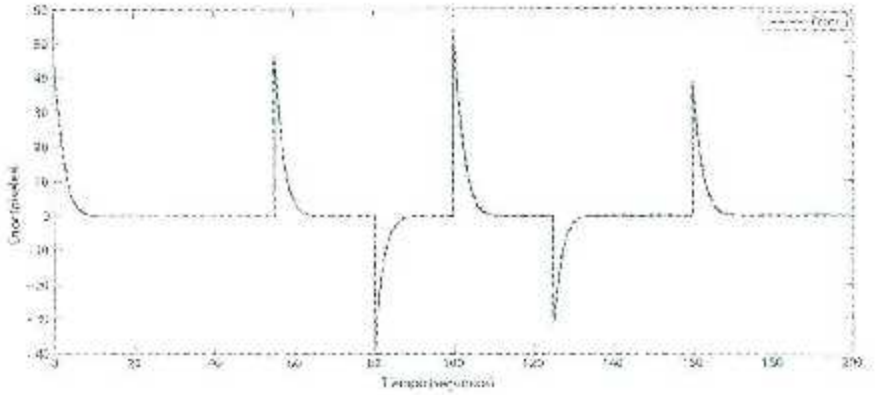

Figura 6.2: Gráfica del error de regulación medido en pixeles

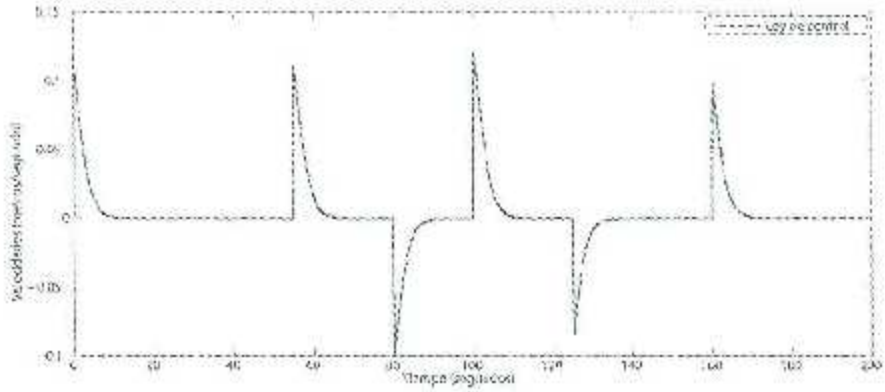

Figura 6.3: Velocidad lateral del carro en regulación (m/s)

simulación es de 100 segundos. El objeto a seguir se sitúa a una distancia z_0 de 6 metros. Los valores de las ganacias para seguimiento se muestran en la tabla 6.3.

Los resultados de la simulación se muestran en las Figuras 6.4, 6.5 y 6.6. En la figura 6.4 se aprecia la señal de entrada (línea continua) y la señal de salida (línea punteada). En la figura 6.5 se presenta un error el cual va disminuyendo conforme avanza el tiempo. Por último en la figura 6.6 se observa la gráfica de las velocidades del carro, en donde la velocidad máxima es de 0.2 m/s (La resolución del carro permite ingresar velocidades de 1 mm/s).

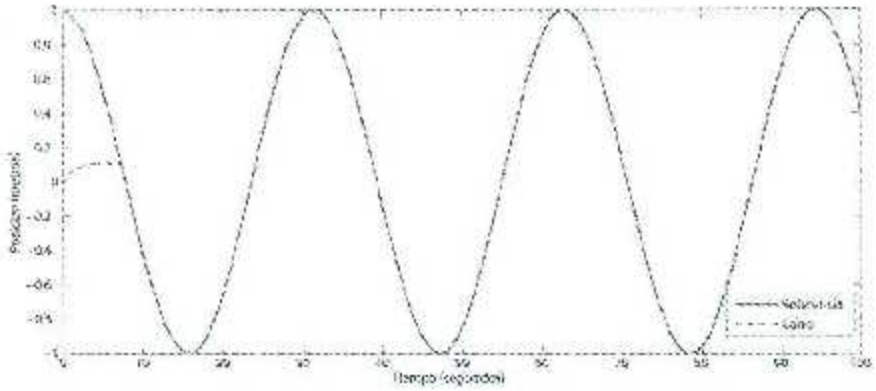

Figura 6.4: Gráficas obtenidas en la simulación de seguimiento

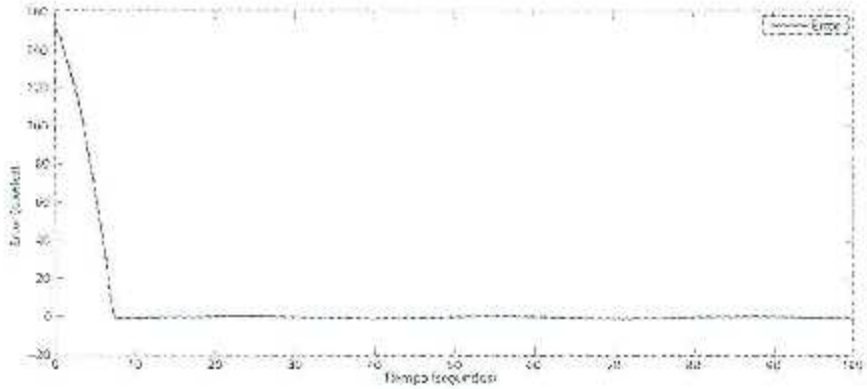

Figura 6.5: Gráfica del error de seguimiento medido en pixeles

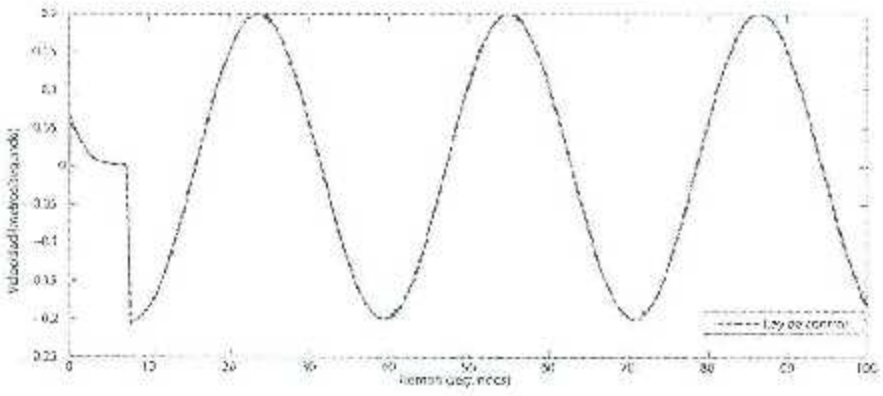

Figura 6.6: Velocidad lateral del carro en seguimiento (m/s)

Tabla 6.2: Parámetros del controlador cinemático en regulación

Parámetros	Valores v_a	Valores v_i
P_{st}	0.25	0.25
k_p	0.03	0.02
P_D	0.25	0.25
k_d	0.001	0.002

Tabla 6.3: Parámetros del controlador cinemático en seguimiento

Parámetros	Valores v_a	Valores v_i
P_{st}	0.5	0.5
k_p	0.9	0.8
P_D	0.5	0.5
k_a	0.2	0.2

6.2. Experimentación

Los experimientos se realizaron dentro del instituto, en un lugar abierto, con suelo firme y sin obstáculos dentro del área de movimiento del robot. El objeto a seguir se muestra en la Figura 6.7 del cual primero se tomó una foto para determinar los valores HSV para la segmentación e interpretación. El círculo blanco que se observa en la Figura 6.7 es introducido en la imagen para identificar el objeto que se está reconociendo. Se busca un comportamiento similar a la simulación de regulación presentada en la Sección 6.1.1. Para esto el objeto se va a desplazar una distancia esperando en esa posicion hasta que el carro se estabilize y nuevamente se mueve el objeto.

Figura 6.7: Objeto utilizado para el experimiento

Se utilizó el programa ARIA para programar el controlador siguiendo los pasos de la

Sección 3.2, el código se presenta en el Aprendice A. Para la obtención de imágenes se emplearon las funciones de la biblioteca V4L2 y el tratamiento se realizó con las funciones de OpenCV 3.2.1.

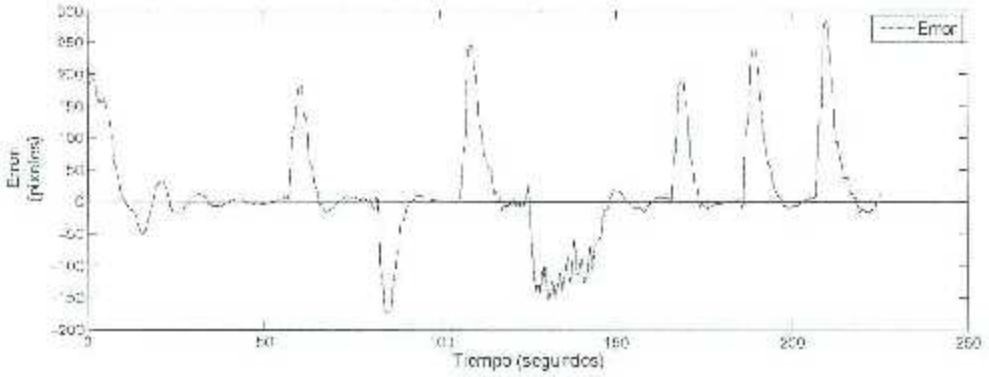

Figura 6.8: Error obtenido en el experimento

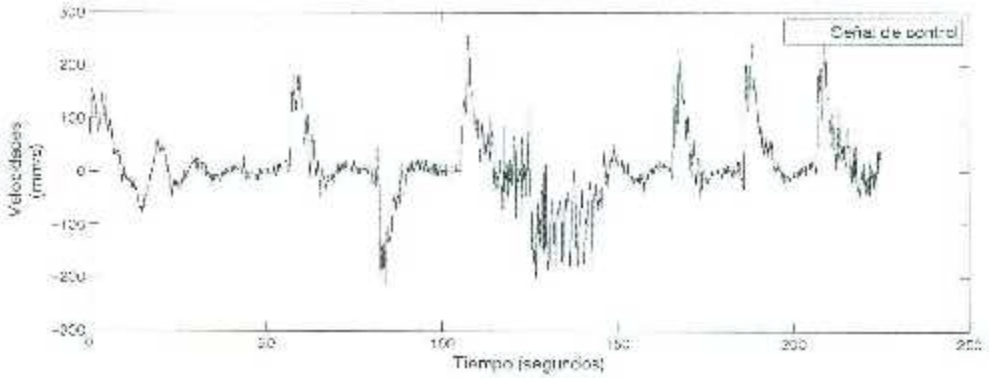

Figura 6.9: Velocidad lateral del robot durante el experimento

Los resultados se muestran en las Figuras 6.8 y 6.9. El experimento consiste en desplazar el carro lo necesario para que el centroide del objeto observado coincida con el eje central de la imagen; una vez centrada la imagen, el objeto es desplazado en forma lateral esperando la reubicación del carro. El tiempo del experimento fue aproximadamente de 4 minutos.

Capítulo 7

Conclusiones

Con el estudio aquí realizado para el robot Seekur, se pudo determinar la capacidad de movimiento del robot. Durante la investigación se encontró que este tipo de robots son muy versátiles y robustos. El Seekur tiene de fábrica dos modos de operación, sin embargo, un robot con tracción independiente tiene la capacidad de tener más modos de operación si se establecen ciertas condiciones. Se puede utilizar como un robot móvil uniciclo si se fijan las cuatro llantas a un mismo ángulo y se usan señales iguales para ruedas de cada lado, o se puede tomar como un robot móvil Ackerman si se dejan fijas las llantas traseras y direccionables las delanteras.

En los experimentos realizados con el control servo-visual basado en imagen, donde la coordenada en el eje de las abscisas se utiliza para el movimiento lateral, los resultados presentados, aunque razonables todavía pueden presentar mejoría agregando una acción integral en el controlador cinemático. Otro punto a considerar para mejorar el seguimiento es la estimación de la distancia a la que se encuentra el objeto, ya que entre más alejado se encuentre el objeto del robot, el movimiento lateral presentará cambios más pequeños debido a que la proyección en el sensor de la cámara tiene un movimiento más lento. Debido a los errores que se producen en el tratamiento de las imágenes, la resolución de las imágenes y la propia naturaleza digital de los datos, se producen una serie de errores en los cálculos. Una ventaja que tiene el robot es el hecho de no presentar restricciones

no holonómicas ya que la trayectoria deseada por más complicada que sea puede ser realizada (cuando opera en el modo omnidireccional). En el presente trabajo, sin embargo, únicamente se toma en cuenta un grado de libertad del robot, por lo que se planea en un futuro usar el avance y orientación del robot para realizar evasión de obstáculos durante el seguimiento del objeto. Los resultados obtenidos dieron suficiente información para presentar un artículo en el congreso 2013 de la Asociación de México de Control Automático (AMCA) [Saenz et al., 2013].

Como trabajo a futuro se planea implementar un controlador discreto con una técnica muy reciente en el área de control llamada "Autómata Celular". Este es un control inteligente y es en tiempo discreto. La razon de utilizar este tipo de control es porque el robot puede recibir órdenes cada 100 milisegundos esperando mejores resultados al aplicar un control en tiempo discreto.

Apéndice A

Código fuente: Control para regulación

```
//-------------------------------------------------------------------------
// LIBRERIAS
//-------------------------------------------------------------------------
\#include ''Aria.h''
\#include <unistd.h>
\#include <stdio.h>
\#include <cv.h>
\#include <highgui.h>
\#include ''sdxcamara.hpp''
\#include ''pide.hpp''
\#include <sys/time.h> //gettimeofday()
using namespace std;
using namespace cv;
//-------------------------------------------------------------------------
// DEFINICION DE VARIABLES
//-------------------------------------------------------------------------
char res;//CARACTER PARA IDENTIFICAR LOS VALORES DE CONVERSACION POR EL TECLADO
string str;
int ret;

//VARIABLES PARA EL PTZ
ArSerialConnection con,conPTZ;
ArPTZ *camara;
ArRobot robot(NULL, false);

//VARIABLES PARA EL CODIGO
//IMAGEN ORIGEN DE LA CAMARA/VIDEO EL TRACKING
IplImage* imgTracking;
```

```
int fps = 20;// [illegible]
int frameW = 640;//[illegible]
int frameH = 480;// [illegible]
double m00 = 0, m01 = 0, m10 = 0;// [illegible]
int posY = 0,posX = 0;//[illegible]
IplImage* frame=0;
CvCapture* capture;

//[illegible]
int lastX = -1;//[illegible]
int lastY = -1;//[illegible]
int ey = 0; //[illegible]
int ex = 0; //[illegible]
double kp1 = 0.0010, //[illegible]
double kp2 = 0.0018; //[illegible]
double u1 = 0;//[illegible]
double u2 = 0;//[illegible]
float tiempo=0;
FILE *bd;
struct timeval comienzo, final;

void *\_\_gxx\_personality\_v0;

//---------------------------------------------------------------------------
// OBTENCION DE HUECOS
//---------------------------------------------------------------------------
IplImage* GetThresholdedImage(IplImage* imgHSV)//[illegible]
{
    IplImage* imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL\_DEPTH\_8U, 1);//[illegible]
    cvInRangeS(imgHSV, cvScalar(27,71,108), cvScalar(62,256,201), imgThresh); //[illegible]
    return imgThresh;//[illegible]
}

void trackObject(IplImage* imgThresh)//[illegible]
{
    CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));//[illegible]
    cvMoments(imgThresh, moments, 1); //[illegible]
    m00 = cvGetCentralMoment(moments, 0, 0);//[illegible]
        m10 = cvGetSpatialMoment(moments, 1, 0);//[illegible]
```

```
    m01 = cvGetSpatialMoment(moments, 0, 1);//OBTENGO M
    if(m00>1000)
        {
        posX = m10/m00;//CALCULA LA POSICION X DEL CONTROL
        posY = m01/m00;//CALCULA LA POSICION Y DEL CONTROL
        if(lastX>-0 && lastY>-0 && posX>-0 && posY>-0)
        {
            cvCircle(imgTracking, cvPoint(posX, posY),20, cvScalar(255,255,255), 4);
        }
        lastX = posX;
        lastY = posY;
    }
    free(moments);
}

int main() //PROGRAMA PRINCIPAL
{
  Aria::init();//INICIAR EL SERVICIO DE ARIA
  if ((ret = con.open()) != 0)//ABRIMOS LA CONEXION CON EL MICRO
  {
    str = con.getOpenMessage(ret);//OBTIENE EL MENSAJE
    printf(''PRUEBA DE CONEXION FALLIDA \%s\textbackslash n'', str.c\_str());//IMPRIME EN
        PANTALLA EL ERROR
    Aria::exit(0);//FINALIZA LA CONEXION CON EL ARIA
  }
robot.setDeviceConnection(&con);//ESTABLECE LA CONEXION CON EL MICRO
if (!robot.blockingConnect())//CREA CONEXION CON ARIA
{
  printf(''NO SE PUDO ESTABLECER COMUNICACION CON EL ROBOT. \textbackslash n'');
  Aria::exit(0);//FINALIZA CONEXION CON EL ARIA
  return 1;
}
//-------------------------------------------------------------------
if(PideEncenderCamara() == SI)
{
      robot.com2Bytes(116,12,1);
}
else
{
      Aria::exit(0);
}
printf(''PROBANDO LA CONEXION DE LA CAMARA\textbackslash n'');
comPTZ.setPort(ArUtil::COM4);
```

```
if(!conPTZ.openSim())
{
   printf(''NO SE PUDO REALIZAR LA CONEXION\textbackslash n'');
   Aria::exit(0);
}
else
{
  printf(''CAMARA CONECTADA AL COM4\textbackslash n'');
}

camara = new ArRVisionPTZ(&robot);
camara->setDeviceConnection(&conPTZ);
printf(''CAMARA INICIADA\textbackslash n'');
camara->init();
robot.setPTZ(camara);

PideGrados(camara);

cout << ''INCIO DE COMUNICACION CON SENSORAY\textbackslash n'';
signal(SIGINT, quit\_handler);
signal(SIGTERM, quit\_handler);

cout << ''CARGANDO PARAMETROS\textbackslash n'';
G\_osd\_text[0] = 0;
G\_jc.quality = 75;
userdata.data = NULL;
userdata.len = 0;
userdata.interval = 0;
type = TYPE\_JPEG;
default_name = ''output.jpg'';
out\_name = out\_name ? out\_name : (char*)default\_name;

      if (strcmp(out\_name, ''-'') == 0)
            G\_fpout = stdout;
      else
            G\_fpout = fopen(out\_name, ''w+'');

      if (G\_fpout == NULL)
      {
            fprintf(stderr, ''invalid filename \%s, exiting\textbackslash n'', out\
                _name);
            exit (EXIT\_FAILURE);
      }
```

```
cout << ''ESTABLECIENDO COMUNICACION\textbackslash n'';
  open\_device ();
cout << ''INICIANDO DISPOSITIVO\textbackslash n'';
  init\_device ();
cout << ''LIMPIANDO BUFFER\textbackslash n'';
  atexit(cleanup);
cout << ''PREPARANDO CAPTURA\textbackslash n'';
  start\_capturing ();
  cout << ''CAPTURAD DE PRUEBA\textbackslash n'';
  mainloop();
  printf(''\textbackslash nPROBANDO CONEXION CON SENSORAY-2253\textbackslash n'');
  capture = cvCaptureFromFile(''output.jpeg'');
  frame = cvQueryFrame(capture);
  if(!frame)
      {
              printf(''ERROR EN LA COMUNICACION. NO SE OBTUVO IMAGEN\textbackslash n'')
                  ;
      }
      else
      {
              printf(''COMUNICACION CON LA CAMARA EXITOSA\textbackslash n'');
      }
      imgTracking=cvCreateImage(cvGetSize(frame),IPL\_DEPTH\_8U, 3);
      cvZero(imgTracking);
      cout << ''CREANDO LA INTERFACES DE OPENCV\textbackslash n'';
      cvNamedWindow(''Video'');
      cvNamedWindow(''Ball'');
      cout << ''PREPARANDO ARCHIVO DE VIDEO \''video.mpeg\''\textbackslash n'';
      VideoWriter ov = VideoWriter( ''video.mpeg'', CV\_FOURCC('P','I','M','1'), fps,
          Size(frameW,frameH));
      //cout << ''LIBERANDO IMAGENES DE PRUEBA\textbackslash n'';
      //imgRed=cvCloneImage(frame);
      //cvReleaseCapture(&capture);
      cout << ''INICIANDO PRIMER CICLO\textbackslash n'';
      while (true)
      {
              cout << ''IMAGEN ADQUIRIDA: '';
              mainloop();
              capture = cvCaptureFromFile(''output.jpeg'');
              frame = cvQueryFrame(capture);
              if(!frame)
              {
                      printf(''PERDIDA DE COMUNICACION. INTENTAR DE NUEVO (R). SALIR(E)
```

```
                ");
            res = getchar();
            if(res=='S'||res=='s')
            {
                    cvReleaseCapture(&capture);
                    robot.com2Bytes(120,12,1);
                    Aria::shutdown();
                    return EXIT\_FAILURE;
            }
            else
            {
                    continue;
            }
        }
        frame=cvCloneImage(frame);

        cvSmooth(frame, frame, CV\_GAUSSIAN,3,3); //smooth the original image
            using Gaussian kernel

        IplImage* imgHSV = cvCreateImage(cvGetSize(frame), IPL\_DEPTH\_8U, 3);
        cvCvtColor(frame, imgHSV, CV\_BGR2HSV); //Change the color format from
            BGR to HSV
        IplImage* imgThresh = GetThresholdedImage(imgHSV);

        cvSmooth(imgThresh, imgThresh, CV\_GAUSSIAN,3,3); //smooth the binary
            image using Gaussian kernel

        //track the position of the ball
        trackObject(imgThresh);

        // Add the tracking image and the frame
        cvAdd(frame, imgTracking, frame);

        //cvShowImage("Ball", imgThresh);
        //cvShowImage("Video", frame);
        //Clean up used images
        cvReleaseImage(&imgHSV);
        cvReleaseImage(&imgThresh);
        cvReleaseImage(&frame);
        cvReleaseCapture(&capture);

        printf("X:\%d,Y:\%d\textbackslash n",5,lastX,5,lastY);
    int c = waitKey(10);
```

```
if((char)c==27 ) break;
 }
        capture = cvCaptureFromFile(''output.jpeg'');
        frame = cvQueryFrame(capture);
        bd = fopen(''controly.txt'',''w'');//ABRIENDO EL ARCHIVO PARA GRABAR LOS
            SMOOTHINGS
        gettimeofday(&comienzo, NULL);
        fprintf(bd,''T:0,X:\%*d,Y:\%*d,&x:\%*d,&y:\%*d,U::\%*.*f,U2:\%*.*f\r\
            textbackslash r'',1,lastX,4,lastY,4,ex,4,ey,5,2,u1,5,2,u2);
 while(true)
 {
//        cout << ''ENTRO'' << endl;
        mainloop ();
//        cout << ''DESPUES IMAGEN.\textbackslash n'';
        capture = cvCaptureFromFile(''output.jpeg'');
//        cout << ''CAPTURO IMAGEN\textbackslash n'';
        frame = cvQueryFrame(capture);
        imgTracking=cvCreateImage(cvGetSize(frame),IPL\_DEPTH\_8U, 3);
        if(!frame)
        {
                printf(''PERDIDA DE COMUNICACION. INTENTAR DE NUEVO (R). SALIR(E)
                    '');
                res = getchar();
                if(res=='E'||res=='e')
                {
                                cvReleaseCapture(&capture);
                                robot.com2Bytes(110,12,1);
                                &rik: shutdown();
                                return EXIT\_FAILURE;
                }
                else
                {
                        continue;
                }
        }
frame=cvCloneImage(frame);

 cvSmooth(frame, frame, CV\_GAUSSIAN,3,3);
//  cout << ''SUAVIZO IMAGEN TEMPORAL'';
 IplImage* imgHSV = cvCreateImage(cvGetSize(frame), IPL\_DEPTH\_8U, 3);//CREACION
     DE IMAGEN TEMPORAL
//  cout << ''CONVERSION A SECUENCIAS'';
 cvCvtColor(frame, imgHSV, CV\_BGR2HSV);//CAMBIA EL ESPACIO DE COLOR DE BGR A HSV
```

```
//      cout << "OBTENIENDO LA IMAGEN HSV" << endl;
        IplImage* imgThresh = GetThresholdedImage(imgHSV);

        cvSmooth(imgThresh, imgThresh, CV_GAUSSIAN,3,3);

        trackObject(imgThresh);
//      cout << "AGREGANDO IMAGENES" << endl;
        cvAdd(frame, imgTracking, frame);
                cv << frame;

        //cvShowImage("Ball", imgThresh);
        //cvShowImage("Video", frame);

                ex = (double)(320 - lastX),
                ey = (double)(240 - lastY);

        u1 = -ex * kp1 + camara->getPan();
        u2 = ey * kp2 + camara->getTilt();

        camara->panTilt(u1,u2);
        gettimeofday(&final, NULL);
        tiempo = (final.tv_sec + (float)final.tv_usec/1000000) (comienzo.tv_sec+(float
            )comienzo.tv_usec/1000000);
        fprintf(bd,"T:\%*.*f,X:\%*d,Y:\%*d,Ex:\%*d,Ey:\%*d,U1:\%*.*f,U2:\%*.*f\r\
            textbackslash n",5,3,tiempo,4,lastX,4,lastY,4,ex,4,ey,5,2,u1,5,2,u2);

//      cout << "LIBERANDO BUFFERS" << endl;
        cvReleaseImage(&imgHSV);
        cvReleaseImage(&imgThresh);
        cvReleaseImage(&frame);
        cvReleaseImage(&imgTracking);
        cvReleaseCapture(&capture);
        int c = waitKey(33);
        if((char)c==27 ) break;
    }
//      cvReleaseImage(&imgTracking);
        cvDestroyAllWindows() ;
        robot.com2Bytes(116,12,0);
        printf("CERRANDO CONEXION CON LA CAMARA");
        conPTZ.close();
        printf("CERRANDO CONEXION CON EL ROBOT");
        fclose(bd);
        Axis::exit(0);
```

```
        return EXIT_SUCCESS;
}
```

Bibliografía

[Arroyo et al., 2012] Arroyo, G., Lara, M., Peralta, R., Gutiérrez, J., Aguirre, J., Gutiérrez, J., Mateos, L., Minarii, Y., and Muñoz, S. (2012). FinDER Robot de búsqueda en ambientes de desastre. In *Congreso Mexicano de Robótica*, pages 1–8, Puebla, Puebla, México.

[Beer et al., 2007] Beer, F. P., Johnston, E. R. J., and Eisenberg, E. R. (2007). *Mecánica Vectorial para Ingenieros: Dinámica*. McGraw-Hill Interamericana.

[Campion et al., 1996] Campion, G., Bastin, G., and Andréa-Novel, B. D. (1996). Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62.

[Canudas et al., 1996] Canudas, C., Siciliano, B., and Bastin, G. (1996). *Theory of Robot Control*. Springer.

[Chang et al., 2010] Chang, F., Zhang, G., Wang, X., and Chen, Z. (2010). PTZ Camera Target Tracking in Large Complex Scenes. In *World Congress on Intelligent Control and Automation*, pages 47–52, Jinan, China.

[Clavien and Fr, 2010] Clavien, L. and Fr, J. (2010). Teleoperation of AZIMUT-3 , an Omnidirectional Non-Holonomic Platform with Steerable Wheels. In *IEE/RSJ International Conference on Intelligent Robots Systems*, pages 2515–2516, Taipei, Taiwan.

[Connette et al., 2008] Connette, C. P., Port, A., Hagele, M., and Verl, A. (2008). Control of an Pseudo-omnidirectional, Non-holonomic, Mobile Robot Based on an ICM Repre-

sentation in Spherical Coordinates. In *IEEE Conference on Decision and Control*, pages 4976–4983, Cancún, México.

[Copot et al., 2010] Copot, C., Burlacu, A., and Lazar, C. (2010). Visual control architecture of servoing systems based on image moments. In *2010 12th International Conference on Optimization of Electrical and Electronic Equipment*, pages 801–806, Basov, Romania.

[Cuevas et al., 2010] Cuevas, E., Zaldívar, D., and Pérez, M. (2010). *Procesamiento digital de imágenes con MATLAB y Simulink*. Alfaomega.

[Dumitrascu et al., 2011a] Dumitrascu, B., Filipescu, A., Minzu, V., Voda, A., and Minca, E. (2011a). Discrete Time Sliding Mode Control of Four Driving-Steering Wheels Autonomous Vehicle. in *Chinese Control Conference*, pages 3620–3625, Yantai, China.

[Dumitrascu et al., 2011b] Dumitrascu, B., Filipescu, A., Vasilache, C., Minca, E., and Filipescu, A. J. (2011b). Discrete-Time Sliding-Mode Control of Four Driving / Steering Wheels Mobile Platform. In *Mediterranean Conference on Control and Automation*, pages 1076–1081, Corfu, Greece.

[Espiau et al., 1992] Espiau, B., Chaumette, F., and Rives, P. (1992). A New Approach to Visual Servoing in Robotics. *IEEE Transactions on Robotics and Automation*, 8(3):14.

[Ferrier, 1998] Ferrier, N. J. (1998). Performance of Visual Tracking Systems : Implications for Visual Controlled Motion. In *IEEE Conference on Decision and Control*, pages 3725–3730, Tampa, Florida.

[García, 2007] García, G. (2007). *Sistema de Control Multisensorial Aplicado a Tareas de Manipulación Automática. Memoria de Suficiencia Investigadora*. Universidad de Alicante, España.

[Hauckt et al., 2000] Hauckt, A., Passigt, G., Schenkf, T., Sorgt, M., and Farbed, G. (2000). On the Performance of a Biologically Motivated Visual Control Strategy for

Robotic Hand-Eye Coordination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1626–1632, Takamatsu, Japón.

[Kane et al., 2010] Kane, G., Boesecke, R., Raczkowsky, J., and Wörn, H. (2010). Kinematic path-following control of a mobile robot on arbitrary surface. In *IEEE International Conference on Robotics and Automation*, pages 1562–1567, Anchorage, Alaska, USA.

[Khalil and Kleinfinger, 1986] Khalil, W. and Kleinfinger, J. (1986). A New Geometric Notation for Open and Close-loop Robots. In *International Conference on Robotics and Automation*, pages 1174 – 1179, San Francisco, California.

[Mao et al., 2012] Mao, S., Huang, X., and Wang, M. (2012). Image Jacobian Matrix Estimation Based on Online Support Vector Regression. *International Journal of Advanced Robotic Systems*, 9:1–9.

[Ollero, 2007] Ollero, A. (2007). *Robótica Manipuladores y Robots Móviles*. Alfaomega.

[Saenz et al., 2013] Saenz, A., Santibañez, V., Ollervides, J., and Dzul, A. (2013). Control servo-visual aplicado a un robot Scekur en un ambiente sin obstáculos : Simulación y Experimentación. In *Congreso Anual de la Asociación de México de Control Automático*, Ensenada, Baja California.

[Siciliano et al., 2009] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics Modelling, Planning and Control*. Springer.

[Solea et al., 2010] Solea, R., Filipescu, A., and Cernega, D. (2010). Lateral Motion Control of Four-Wheels Steering Vehicle Using a Sliding-Mode Controller. In *Chinese Control Conference*, pages 3699–3703, Beijing, China.

[Swain et al., 1998] Swain, R., Devy, M., and Jonquières, S. (1998). Navegación de un Robot Móvil por Medio de Control Visual en Ambiente Estructurado. *Computación y Sistemas*, 1(3):9.

[Yoshida and Tsuzuki, 2006] Yoshida, Y. and Tsuzuki, K. (2006). Visual Tracking and Control of a Moving Overhead Crane Load. In *IEEE International Workshop on Advanced Motion Control*, pages 630–635, Istanbul, Turquía.