UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

SEGMENTACIÓN ROBUSTA DE IMÁGENES A COLOR APLICADA A UN ROBOT DE SERVICIO

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Oscar Alejandro Vitela Ramírez

Dr. Félix Calderón Solorio

Director de Tesis

Leonardo Romero Muñoz

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Agosto 2018

Morelia, Michoacán

SEGMENTACIÓN ROBUSTA DE IMÁGENES A COLOR APLICADA A UN ROBOT DE SERVICIO

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***Oscar Alejandro Vitela Ramírez.***

Dr. Juan José Flores Romero
Presidente del Jurado

Dr. Félix Calderón Solorio
Director de Tesis

Dr. Leonardo Romero Muñoz
Co-Director

Dr. Jaime Cerda Jacobo
Vocal

Dr. Carlos Alberto Júnez Ferreira
Revisor Externo (Fac, Ing, Civil UMSNH)

Dr. Félix Calderón Solorio
Jefe de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica. UMSNH (Por reconocimiento de firmas).

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Agosto 2018

Resumen

La segmentación de imágenes de color es una área importante en la visión computacional y es ampliamente utilizada en muchos campos de aplicación. Esta segmentación le permite a un robot móvil conocer la existencia y ubicación de determinados objetos de interés que se encuentran dentro de su ambiente. La tarea de segmentar una imagen a color en un robot móvil debe considerar, entre otros factores, los cambios de iluminación en el ambiente y la diferente percepción del objeto por una cámara, dependiendo de la orientación del objeto y la distancia del objeto a la cámara.

En esta tesis, se presenta un método de segmentación de imágenes a color robusto ante cambios de iluminación y de ubicación de objetos. El método propone un modelo bayesiano de aprendizaje supervisado que involucra la maximización de funciones discriminantes basadas en los histogramas de color de las imágenes, agregando restricciones de coherencia espacial. La coherencia espacial incluye, formalmente en el modelo de optimización, la idea intuitiva de que pixeles adyacentes de la imagen muy probablemente corresponden al mismo objeto. El método de segmentación propuesto se integró en un pequeño robot móvil equipado con una cámara y una pinza, capaz de recoger pelotas de colores y transportarlas a pequeños cestos o depósitos. Para que el robot pudiera realizar la tarea, se desarrolló una aplicación en el lenguaje C++ utilizando la biblioteca OpenCV. Se abordó la adquisición de las imágenes, calibración de la cámara, cálculos de los histogramas de color, obtención de una vista superior (vista de pájaro) de los objetos (a partir la imagen obtenida de la cámara), etc.

Se realizaron experimentos para valorar el desempeño del robot móvil en la tarea de recolección de pelotas de color utilizando la segmentación propuesta y la aplicación desarrollada. Los resultados obtenidos muestran mejores desempeños tanto en la segmentación de color como en el tiempo utilizado por el robot móvil, comparado con trabajos previos realizados por estudiantes de maestría. El modelo bayesiano propuesto, basado en histogramas e incorporando la coherencia espacial, explica los buenos resultados en la parte de segmentación de las imágenes a color. Por otro lado, la calibración cuidadosa de la cámara y la generación de una vista superior de los objetos, permitió ubicar con precisión la ubicación de los objetos en el piso, reduciendo el tiempo requerido por el robot para dirigirse a los objetos y a los cestos.

Palabras Clave: Segmentación de color, robot de servicio, funciones discriminantes, coherencia espacial, vista de pájaro, encontrar cúmulos.

Abstract

Color image segmentation is a very important field in computational vision and is widely used in many application fields. This segmentation allows a mobile robot to know the existence and position of determined objects of interest that are inside the environment of the robot. The task of segmenting a color image in a mobile robot must consider, lighting changes in the environment and the diferent perception of an object for the camera, depending on the object orientation and the distance to the object.

In this thesis, we present a robust segmentation method for color image to lighting changes and object location. The method propose a bayesian model of supervised learning that involves maximization of discriminant functions based on color histograms of the images, adding restrictions of spatial coherence. Spatial coherence includes formaly in the optimization model the idea that adjacent pixels of the image are more likely to belong to the same object. The segmentation method was integrated to a mobile robot equiped with a camera and a gripper, it is capable of gather color balls and take them to little deposits. To get the robot to accomplish its task, we developed an application in C++ language with the OpenCV library. Between other tasks, we abord the acquisition of images, camera calibration, color histogram calculation, calculation of an upper view (bird's eye) of the objects (from the camera image), etc.

We made experiments to check the mobile robot performance in the task of gather color balls using the segmentation proposed and the developed application. The results obtained show a better perfomance in segmentation and robot time compared to a previous work made by master degree students. The bayesian model proposed, based on histograms and adding the spatial coherence explain the better results obtained in the segmentation part of the color images. Meanwhile, the camera calibration and generation of an upper view of the objects, allowed the robot to locate the objects on the floor with precision, reducing the time required for the robot to go to the objects and deposits.

Keywords: Color segmentation, service robot, discriminant functions, spatial coherence, bird's eyes, finding cumulus.

Contenido

Lista de Figuras

Lista de Tablas

Acrónimos

SMDF	Segmentation by Maximization of Discriminant Functions.
EM	Expectation Maximization.
RGB	Red-Green-Blue.
HSV	Hue-Saturation-Value.
SRG	Seeded Region Growing.

Lista de Algoritmos

Lista de Símbolos

I	Matriz correspondiente a una imagen.
$I(n,m)$	Pixel correspondiente a la imagen I en la posición (n,m).
$I_R(n,m)$	Cantidad de color rojo para el pixel de la imagen I en la posición (n,m).
$I_G(n,m)$	Cantidad de color verde para el pixel de la imagen I en la posición (n,m).
$I_B(n,m)$	Cantidad de color azul para el pixel de la imagen I en la posición (n,m).
n_r	Número de renglones de una imagen.
n_c	Número de columnas de una imagen.
$x(n,m)$	Vector de características para un pixel en la posición (n,m).
w_i	Clase de color i.
C	Número de clases.
$P(w_i)$	Probabilidad para la clase w_i.
$P(x)$	Probabilidad para el vector de características x.
$P(x\|w_i)$	Probabilidad condicional de que ocurra x dada la clase w_i.
$P(w_i\|x)$	Probabilidad condicional de que ocurra la clase w_i dado x.
$T(A,B)$	Función de índice de Tanimoto.
$q(n,m)$	Etiqueta que representa la clase de color a la cual pertenece el pixel $I(n,m)$.
$f_i(x(n,m))$	Función discriminante para la clase i.
Φ	Conjunto de clases.
$F(\hat{q}(n,m))$	Función propuesta para maximización.
d	Valor entero positivo para definir una ventana de tamaño $(2d+1)\times(2d+1)$.
$S_i(n,m)$	Sumatoria de funciones discriminantes dentro de una ventana para la clase i.
$\bar{q}_{n,m}$	Valor constante para una ventana centrada en (n,m).
$F_s(\bar{q}_{n,m})$	Función propuesta para maximización con coherencia espacial.
$G_i(K,L)$	Función propuesta para calcular imagenes incrementales para la clase i.
$h_i[x]$	Histograma para el vector x en la clase i.
H_i	Constante de normalización para el histograma h_i.
Λ	Matriz intrínseca de la cámara.
$(\hat{x},\hat{y},\hat{z})$	Coordenadas del ambiente real del robot.
$\phi_{\hat{x}}$	Distancia focal de la cámara en el eje $\hat{x}$.
$\phi_{\hat{y}}$	Distancia focal de la cámara en el eje $\hat{y}$.

$\delta_{\hat{x}}$	Centro óptico de la cámara en el eje $\hat{x}$.
$\delta_{\hat{y}}$	Centro óptico de la cámara en el eje $\hat{y}$.
γ	Parámetro de desviación del plano de la imagen en la cámara.
Ω	Matriz de rotación de la cámara.
τ	Vector de traslación de la cámara.
β_1	Primer parámetro de distorción radial de la cámara.
β_2	Segundo parámetro de distorción radial de la cámara.
D	Conjunto de colas utilizado para encontrar cúmulos.
D_k	Cola utilizada para almacenar puntos de un cúmulo.
M	Conjunto utilizado para almacenar cúmulos.
M_k	Conjunto de puntos pertenecientes a un cúmulo.
$\bar{r}_k$	Centro de masa para el cúmulo k en los renglones de la imagen.
$\bar{c}_k$	Centro de masa para el cúmulo k en las columnas de la imagen.
Σ_k	Matriz de covarianza correspondiente al cúmulo k.

Capítulo 1

Introducción

La segmentación de color tiene una gran variedad de aplicaciones en el mundo real ya que es ampliamente utilizada en campos como la medicina, reconocimiento de rostro e iris, e incluso en la detección de transeuntes, por mencionar algunas [Yuheng17]. También es ampliamente utilizada en el mundo de la robótica. Actualmente existen robots que utilizan la segmentación de color como una técnica para llevar a cabo tareas interesantes, ya sea recoger objetos (por ejemplo, una manzana), esquivar obstáculos, entre otros [Wasik02].

Si el objetivo de la segmentación es funcionar en tiempo real, capturando varias imágenes de una cámara por segundo, se puede presentar el problema de que la segmentación falle debido a cambios de iluminación en el ambiente. En este tipo de entornos, algunos métodos como la segmentación basada en umbrales ha mostrado ser poco eficiente, por lo que es necesario buscar otros enfoques [Wasik02].

En esta tesis se aborda el problema de segmentación de imágenes a color, buscando lograr una segmentación eficaz y precisa a pesar de los cambios de iluminación del ambiente y que al mismo tiempo sea lo suficientemente rápida para poder ser realizada en tiempo real por un robot de servicio.

El modelo de segmentación será implementado en un robot de servicio, el cual fue construido para ser utilizado en una competencia de robots de servicio que consiste en recolectar pequeñas pelotas de color rojo y azul y llevarlas a un depósito de su mismo color [com16]. Este robot tiene una cámara Logitech modelo C920, fue construido con

motores de pasos que le permiten avanzar una distancia específica en milímetros y cuenta con una pinza que le permite recolectar las pelotas. Los detalles técnicos del robot y su programación serán descritos en el capítulo 4.

1.1. Planteamiento del problema

Durante el desarrollo de esta tesis, se pretende resolver dos problemas principales. Por un lado, se plantea el problema de la segmentación robusta de imágenes de color ante cambios de iluminación en el ambiente. Por otro lado, se plantea el problema de programar un robot de servicio que pueda realizar las tareas de la competencia de robots de servicio [com16].

1.1.1. Segmentación robusta de imágenes de color

La segmentación de imágenes de color es una área fundamental en el campo de la robótica, principalmente cuando es necesario que el robot deba identificar objetos del ambiente a través de su color. Existen muchos modelos de segmentación de imágenes de color, sin embargo, varios de estos modelos son poco eficientes cuando existen cambios de iluminación en el ambiente. A continuación se presentan las siguientes problemáticas de segmentación de color resueltas en esta tesis:

- La segmentación de color aplicada en tiempo real puede fallar si existen cambios de iluminación en el entorno en el que se trabaja [Wasik02].
- Algunos métodos de segmentación como la segmentación basada en umbrales pueden no ser lo suficientemente precisos cuando existen cambios de iluminación, entregando resultados poco satisfactorios [Wasik02].
- Algunos modelos de segmentación que son robustos ante cambios de iluminación son tardados o consumen muchos recursos de computadora, lo cual los vuelve poco aptos para realizar su tarea en tiempo real [Wasik02].

En la Figura 1.1 se muestra el problema de iluminación, en (a) se muestra una imagen de un vaso de color azul, en (b) se muestra la otra imagen del mismo vaso sometido

(a) **(b)** **(c)**

Figura 1.1: Ejemplo del problema de iluminación. En a) y b) se muestra una imagen del mismo vaso bajo diferentes tipos de iluminación. En c) se muestra el histograma correspondiente al canal B (en el espacio de color RGB) de ambas imágenes, donde el histograma azul representa los pixeles del vaso en a), y el histograma rojo representa los pixeles del vaso en b).

a una iluminación diferente, se puede observar para ambas imágenes una gran diferencia de color (al menos para el ojo humano), en (c) se muestra el histograma correspondiente a los pixeles de ambos vasos para el canal azul (B) en el espacio de color RGB. Se puede observar que la distribución del color azul es diferente para ambas imágenes, a pesar de que provienen del mismo objeto. Esto presenta un problema para algunos tipos de segmentadores, como en la segmentación basada en umbrales [Wasik02].

1.1.2. Robot de servicio

Durante el desarrollo de esta tesis, se pretende resolver el problema de programar un robot de servicio el cual fue construido para poder participar en una competencia de robots de servicio [com16]. Las problemáticas que este robot debe resolver para dicha competencia son las siguientes:

- El robot debe ser capaz de buscar e identificar las pelotas rojas y azules a través de su color y forma.
- El robot debe ser capaz de buscar e identificar los depósitos a través de su color y forma.
- El robot debe ser capaz de desplazarse hacia las pelotas de color y tomarlas con su

pinza.

- El robot debe ser capaz de desplazarse hacia los depósitos y soltar la pelota en el depósito del mismo color que la pelota.

De acuerdo con el concurso de robots de servicio [com16], el tiempo y el número de pelotas recogidas son los factores que se toman en cuenta para determinar el robot ganador de la competencia. Por lo tanto, el robot de servicio debe terminar la tarea de recoger todas las pelotas en el menor tiempo posible.

1.2. Antecedentes

Llevar a cabo una segmentación robusta de color es un problema que ya tiene múltiples propuestas de solución. Algunos de los enfoques que se han propuesto consisten en una mezcla de técnicas convencionales como segmentación basada en umbrales, métodos basados en bordes y métodos basados en regiones por mencionar algunos. Estos métodos son explicados a continuación.

- **Segmentación de color basada en umbrales:** Cuando un pixel cuyo valor de color se encuentra dentro de dos umbrales dados, donde cada umbral se define como un valor dentro de un intervalo $[a, b]$, entonces se asume que este pixel pertenece a una clase de color dada. Para el caso de una imagen a color, se utiliza un intervalo para cada canal de color del pixel dado, usualmente estos umbrales son fijos y son extremadamente sensibles a los cambios de iluminación [Wasik02].

- **Segmentación de color basada en Bordes:** Este método asume que el valor de color de los pixeles cambia de manera repentina en el borde que existe entre dos regiones de la imagen. Algunos detectores de bordes como Sobel, Canny y Susan [Akram13] son usados en conjunto con algunos métodos post-procesamiento para obtener una detección de bordes de calidad, esto provoca que esta segmentación sea costosa en términos de desempeño computacional [Wasik02].

- **Segmentación de color basada en regiones:** Este método asume que los pixeles que se encuentran dentro de una región de una imagen tienen valores de color similares,

esta similitud depende de un criterio de homogeneidad. Si los valores de color de una región son similares a otra, ambas regiones son mezcladas formando una sola región. Uno de los métodos de este tipo más popular es el crecimiento de regiones con semillas (Seeded Region Growing o SRG) [Adams94] el cual utiliza pixeles llamados semillas, los cuales son usados para definir el valor de color promedio de una región [Wasik02].

- **Segmentación de color utilizando el modelo Numford-Shah:** Consiste en formular la segmentación como un problema de minimización, una imagen de entrada se modela como una función a trozos la cual es llamada como la funcional de Numford-Shah. La funcional busca reforzar las siguientes condiciones de la imagen: 1) El modelo de la imagen debe ser similar a la imagen de entrada; 2) El modelo de la imagen debe tener regiones suaves, excepto en los bordes; 3) Deben existir pocos bordes en el modelo de la imagen [Mumford85].

El tema de poder llevar a cabo segmentación de color robusta ante cambios de iluminación en el mejor tiempo posible y ahorrando recursos de memoria aún es caso de estudio [Wasik02]. A continuación se presentan algunos de los métodos que se han propuesto para resolver este problema.

1.2.1. Transformación adaptativa de distribución de color

Luca Iocchi [Iocchi06] propuso un método conocido como tranformación adaptativa de distribución de color. Este método consiste en emplear transformaciones dinámicas del espacio de color de una imagen, para una imagen dada se calcula su distribución de color en el canal H del espacio de color HSV, en este espacio, H representa el tono de color (en inglés *Hue*), S representa la saturación y V representa el valor de color desde el negro hasta el tono de color puro, luego se seleccionan umbrales fijos para aplicar segmentación de color con umbrales para esta distribución. El problema con estos umbrales es que cuando la distribución de color de la imagen sea diferente a causa de los cambios de iluminación, los umbrales seleccionados ya no serán útiles para conseguir una segmentación precisa.

En este modelo se define una función de transformación la cual recibe una distribución de color y devuelve una distribución de color nueva, el objetivo de esta transformación

es modificar la distribución de color de una imagen de manera que los umbrales fijos previamente seleccionados puedan seguir funcionando para obtener una segmentación precisa a pesar de los cambios de iluminación [Iocchi06].

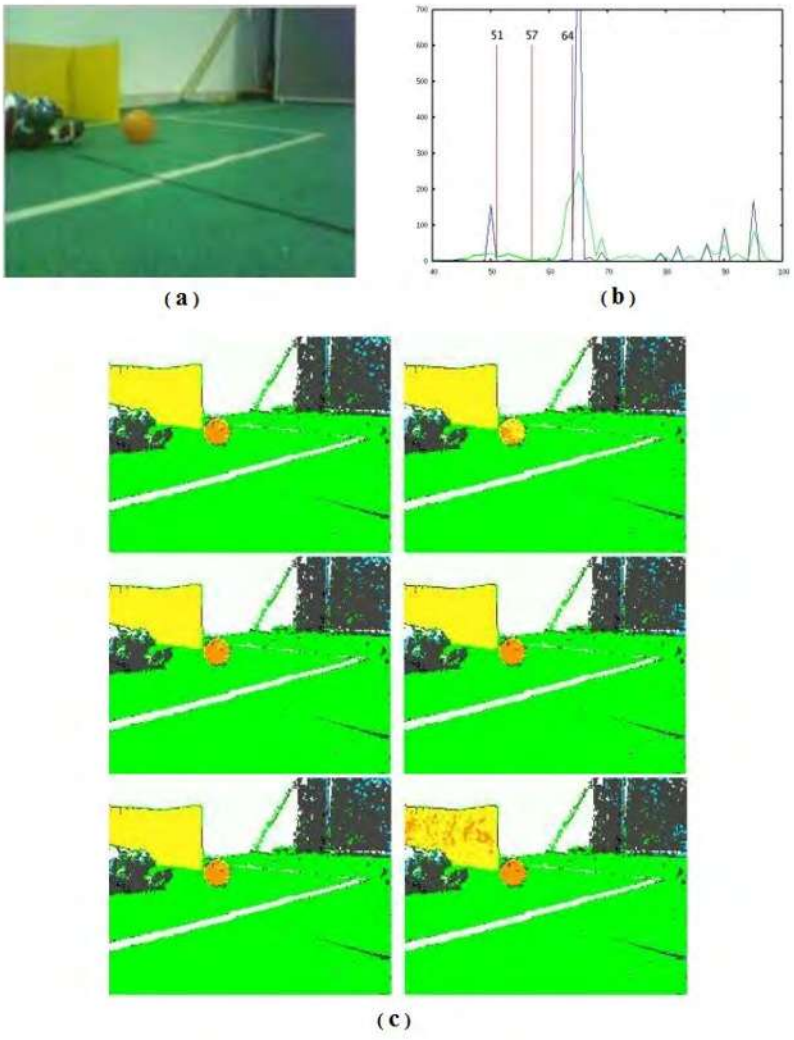


Figura 1.2: Aplicación del algoritmo de Transformación adaptativa a una imagen a), en b) se muestra la distribución calculada en color verde, en azul la distribución calculada a través de la transformación, y en rojo las líneas verticales que definen los 3 umbrales seleccionados para aplicar la segmentación. En c) se muestran los resultados de segmentación (fuente: [Iocchi06]).

En la Figura 1.2 se muestra un resultado de esta segmentación, en a) se muestra la imagen original, en b) se muestra la distribución de la imagen en color verde, en azul

la distribución calculada a través de la transformación, y en rojo las líneas verticales que definen los 3 umbrales para los tonos de color naranja-amarillo, en c) se muestran los resultados donde en el lado izquierdo se muestra la segmentación aplicando la transformación, mientras que en el lado derecho se muestra la segmentación con la distribución original, se puede observar en c) que para las imágenes mostradas del lado izquierdo, la mayoria de los pixeles correspondientes a la pelota de la imagen son de color naranja, lo cual facilita la segmentación de la misma basada en umbrales [Iocchi06].

La ventaja de este enfoque es que las distribuciones de color pueden ser recalculadas aplicando la transformación cada vez que se recibe una imagen nueva, permitiéndole ser usado en tiempo real. Los resultados demostraron que el método es robusto ante cambios de iluminación en comparación con métodos de segmentación basados en umbrales fijos.

1.2.2. Crecimiento de regiones con semillas mezclado con umbrales

Zbigniew Wasik [Wasik02] propuso un método híbrido que integra segmentación con umbrales y crecimiento de regiones con semillas. Consiste en calcular un conjunto de semillas iniciales las cuales son calculadas a través de segmentación por umbrales, para cada una de las clases de color.

El algoritmo fue adaptado para utilizar múltiples semillas iniciales. Estas semillas son empleadas para generar un incremento de regiones de color y formar manchas (en inglés *blobs*) a partir de ellos. Por último las manchas que se encuentren juntas y tengan un valor de color similar, son unidas para formar una sola región. En la Figura 1.3 se muestra la unión de manchas una vez realizado el incremento de regiones.

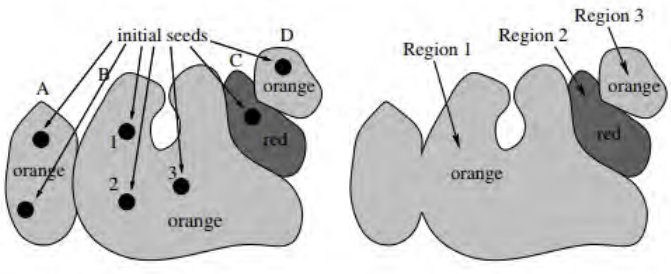
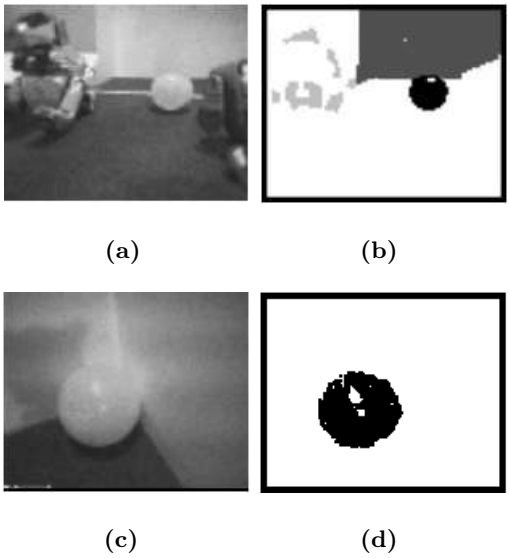


Figura 1.3: Unión de manchas utilizando SRG (fuente: [Wasik02]).

En la Figura 1.4 se pueden observar los resultados de este método, dichos resultados fueron comparados con técnicas de segmentación basadas en umbrales, dando mejores resultados.



(a) **(b)**

(c) **(d)**

Figura 1.4: Segmentación aplicada con el método de crecimiento de regiones con semillas mezclada con umbrales. En (a) y (c) se muestran las imágenes originales, en (b) y (d) se muestran los resultados de la segmentación (fuente: [Wasik02]).

1.2.3. Segmentación de color aplicando EM e Histogramas

Adrián González [Gonzalez09] propone un método el cual consiste en mezclar dos técnicas, por un lado aplicar un método basado en el algoritmo Expectation-Maximization (EM) [Dempster77], con el cual estima los parámetros para poder calcular la probabilidad de pertenencia al conjunto de clases de colores que se tiene, esto para cada uno de los pixeles de la imagen, de esta manera se obtiene una matriz de las mismas dimensiones que la imagen donde cada elemento de la matriz corresponde a un valor de probabilidad entre 0 y 1, teniendo así una matriz de probabilidades para cada clase. El método se apoya en una serie de patrones de entrenamiento.

Por otro lado se calcula la correspondencia del objeto a través del cálculo del histograma en el canal Hue y la aplicación de máscaras en saturación y luminancia. Esto permite hacer un filtrado de la imagen con respecto a la matriz de probabilidades calculada

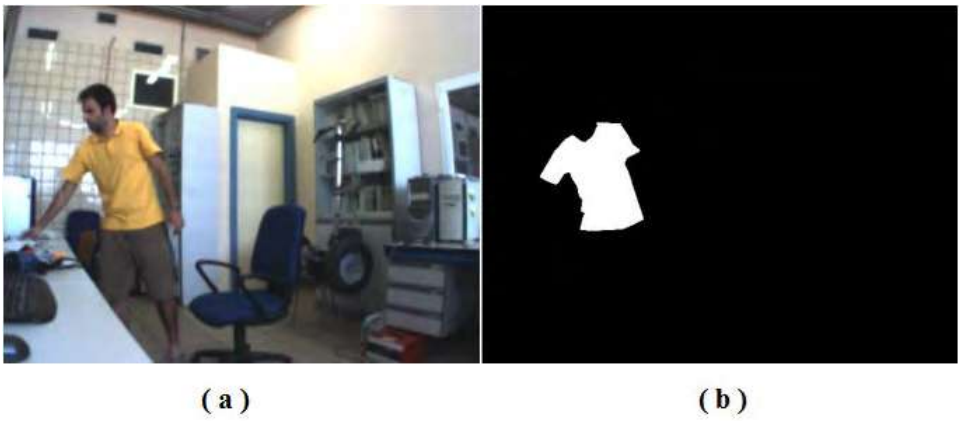


Figura 1.5: Aplicación del algoritmo con EM e Histogramas. En (a) se muestra la imagen original, en (b) se muestra el resultado de la segmentación (fuente: [Gonzalez09]).

con el algoritmo EM, haciendo que el método sea robusto ante cambios de iluminación [Gonzalez09].

En la Figura 1.5 se observan los resultados de aplicar el método. De acuerdo con los resultados, el algoritmo funciona de manera robusta, siempre y cuando no existan cambios bruscos en el ambiente o de forma en el objeto que se quiere segmentar.

1.2.4. Segmentación de color para un robot de servicio que clasifica objetos y los coloca en un contenedor

Garnica et al. [Garnica16] describen un método para resolver el problema del robot de servicio que se aborda en esta tesis. Su método consiste en calcular las probabilidades de cada pixel. Dado que el método procesa imágenes a color, cada pixel está dado por los canales R, G, B donde cada canal está dado por un valor. Después se determina la proporción de rojo a verde y de azul a verde para cada pixel, esta proporción consiste en dividir el valor del canal verde entre el valor del canal rojo o azul y es calculada para cada una de las clases que se quieren reconocer. Con las proporciones de color se obtiene la probabilidad de que el pixel pertenezca a una distribución de probabilidad Gaussiana bidimensional con varianza igual a 9 y covarianza igual a 0, el cálculo de la proporción rojo a verde y azul a verde son utilizados para calcular la media de cada gaussiana, por último se determina que el pixel pertenece a la clase para la cual se obtuvo la mayor probabilidad.

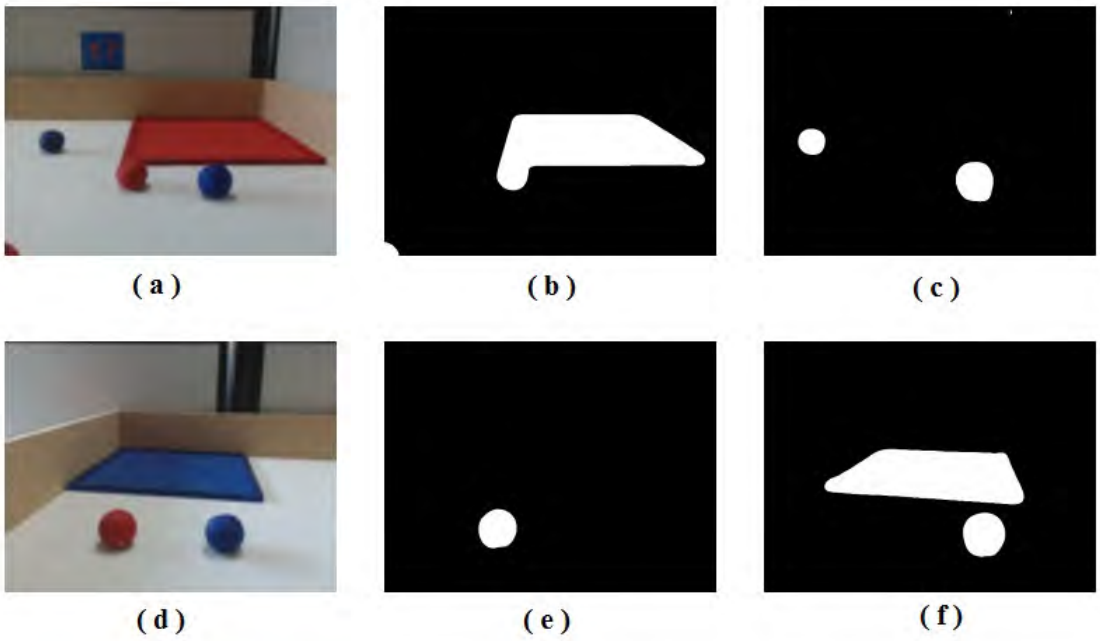


Figura 1.6: Resultados del algoritmo propuesto en [Garnica16]. En (a) y (d) se muestra la imagen original. En (b) y (e) se muestran los resultados de la segmentación para el color rojo. En (c) y (f) se muestran los resultados de la segmentación para el color azul.

En la Figura 1.6 se muestran los resultados de esta segmentación, se puede observar que para las imágenes originales (a) y (d), en sus respectivas máscaras de rojo se detectaron de forma correcta tanto las pelotas como el tablero rojo, el mismo caso ocurre para la máscara azul, esto es una prueba de que la segmentación entregó resultados positivos. Por otro lado, la segmentación permitió al robot de servicio realizar su tarea de forma adecuada en ambientes controlados. Sin embargo, como se menciona en la conclusión de dicho artículo los cambios de iluminación en el ambiente afectan en gran medida la identificación de los colores y repercute negativamente en la segmentación, por lo cual no se considera un método robusto ante cambios de iluminación [Garnica16].

En esta tesis se propone un método de segmentación basado en un modelo probabilístico el cual permite lograr una segmentación lo suficientemente robusta ante los cambios de iluminación del ambiente, posteriormente, esta segmentación será aplicada a un robot de servicio y se compararán los resultados con el método propuesto por Garnica et al. [Garnica16].

1.3. Objetivos de la tesis

1.3.1. Objetivo general

El objetivo de esta tesis es desarrollar un modelo robusto de segmentación de imágenes a color que pueda funcionar de manera eficiente a pesar de los cambios de iluminación que se generen en un ambiente dado. Dicho método debe ser rápido para ser aplicado en un robot de servicio.

Por otro lado, se tiene como objetivo desarrollar una aplicación que le permita al robot cumplir con las tareas dadas por la competencia de robots de servicio mostrado en [com16], estos modelos permiten obtener información del ambiente la cual le permite al robot saber cuanto debe desplazarse para tomar una pelota y llevarla a un depósito, el robot debe llevar a cabo esta tarea en el menor tiempo posible.

1.3.2. Objetivos particulares

A continuación se presentan los objetivos particulares, los cuales son separados en dos partes, por un lado se muestran los objetivos que debe cumplir el modelo de segmentación de color, y por otro lado se muestran los objetivos que debe cumplir el robot de servicio que será programado.

- El modelo de segmentación debe ser capaz de cumplir los siguientes objetivos:
  - Debe ser robusto a los cambios de iluminación de un ambiente dado.
  - Debe ser lo suficientemente rápido para ser utilizado en tiempo real.
  - Debe consumir pocos recursos de computadora para ser aplicado en la robótica.
  - Debe ser lo más exacto posible, esto con el fin de poder utilizar la información proporcionada por la segmentación para implementar algoritmos que permitan a un robot desplazarse de forma precisa en su ambiente de trabajo. En el capítulo 4 se darán más detalles con respecto a este tema.
- La programación del robot de servicio debe cumplir con los siguientes objetivos:

- La cámara del robot debe pasar por un proceso de calibración, donde se espera obtener los parámetros intrínsecos y de distorsión de la cámara, haciendo la corrección de distorsión radial sobre la cámara del robot. Este procedimiento permitirá al robot realizar cálculos de la ubicación de los objetos contenidos en las imágenes de la cámara, con precisión milimétrica.
- Se debe aplicar una transformación de vista de pájaro, la cual permitirá al robot tener información de la distancia a la que se encuentran los objetos que están siendo reconocidos.
- El robot debe ser capaz de tomar decisiones, como buscar y recoger una pelota, y llevarla a un depósito.
- El robot debe ser capaz de distinguir la forma de los objetos que está reconociendo, esto para que el robot pueda saber cuando está viendo una pelota, o un depósito.

1.4. Descripción de capítulos

En el capítulo 2 se presentan las bases para generar modelos de segmentación probabilísticos, presentando un segmentador Bayesiano. En el capítulo 3 se presenta el algoritmo de Segmentación por maximización de funciones discriminantes (SMDF), también se presenta una extensión de este algoritmo permitiéndole funcionar para varias clases. En el capítulo 4 se presenta la programación del robot. En el capítulo 5 se presentan los experimentos y resultados que se realizaron con los algoritmos y el robot de servicio. Por último, en el capítulo 6 se presentan las conclusiones, así como los trabajos futuros relacionados con esta tesis.

Capítulo 2

Segmentación

En este capítulo se presentan las bases para una segmentación probabilística, las cuales son esenciales para proponer técnicas y enfoques que nos permitan realizar una segmentación robusta ante posibles cambios de iluminación en el ambiente.

2.1. Introducción

David Forsyth define la segmentación como el proceso de particionar una imagen en un conjunto de regiones donde cada región representa una área de interés. En la Figura 2.1, se muestra un ejemplo de segmentación, donde se busca como objetivo separar el avión del cielo [Forsyth11].

Simon Prince define a la segmentación como el proceso de asignar una etiqueta a cada uno de los pixeles de la imagen, de forma que las regiones que pertenecen al mismo objeto sean asignadas a la misma etiqueta. Para esto, se define un conjunto de etiquetas $a = \{a_0, a_1, ..., a_N\}$ donde N es el número de pixeles en la imagen dada. Cada etiqueta a_i toma uno de los valores de las C clases $a_i = \{0, 1, ..., C\}$ y ésta es asignada al pixel correspondiente en la posición i [Prince12].

La segmentación puede ser considerada como un problema de clasificación donde se tiene como objetivo asignar un pixel de la imagen a un grupo o clase. Si definimos un pixel de la imagen como un vector de características x, queremos asignar dicho vector a un grupo o una clase basándonos en las características que este contiene. Entonces, si tenemos

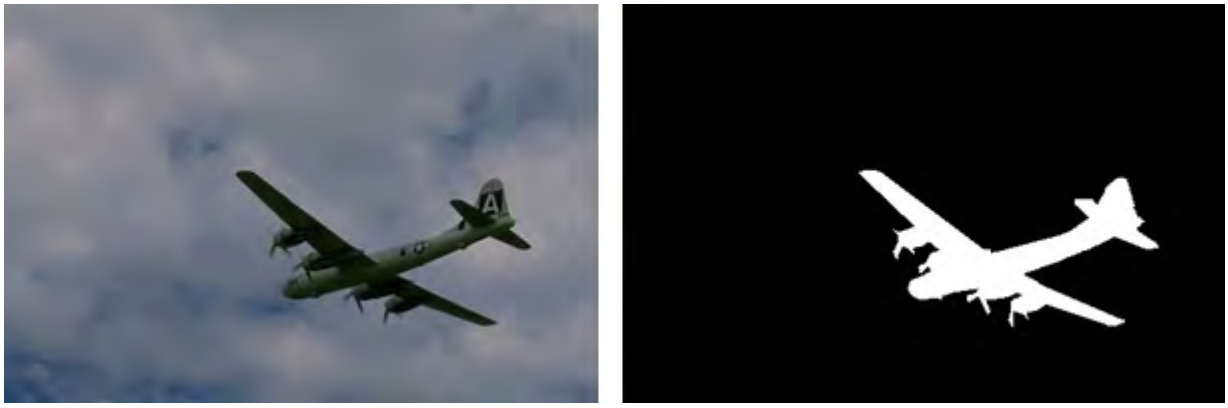


Figura 2.1: Ejemplo de segmentación [Khan14].

un conjunto de C clases $w_0, w_1, w_2, ..., w_{C-1}$, queremos asignar el vector x a la clase a la que pertenezca.

Un enfoque para determinar a que clase podemos asignar el vector x, es calculando la probabilidad de que este vector pertenezca a cada una de las clases. De esta manera, la probabilidad $P(w_i|x)$ que sea mayor representará que el vector x pertenece a la clase w_i.

Los modelos probabilísticos usualmente se apoyan en un conjunto de entrenamiento, el cual consiste en un conjunto de patrones para los cuales se conoce la clase a la que estos pertenecen. Utilizando estos datos, se pueden ajustar parámetros de distribuciones de probabilidad (por ejemplo, media y varianza), los cuales son útiles para calcular la probabilidad de que el vector x pertenezca a una clase. Este procedimiento es conocido como aprendizaje supervisado [Bishop06].

2.2. Segmentación Bayesiana

El teorema de Bayes es la base fundamental en la teoría de decisión Bayesiana, la cual permite ver el problema de clasificación desde un punto de vista probabilístico. Dado que la segmentación de color puede ser expresada como un problema de clasificación, el teorema de Bayes puede ser utilizado [Duda73].

La razón por la que queremos conocer el clasificador Bayesiano es el hecho de que es bastante simple y fácil de implementar, también es rápido ya que solo requiere

de pocos cálculos matemáticos, puede ser utilizado tanto para problemas de clasificación binaria y multi-clase y puede trabajar con características continuas y discretas. Las bases de probabilidad para entender el teorema de Bayes pueden verse en [Freund00].

Dado que tenemos un vector x y un conjunto de clases, podemos aplicar el teorema de Bayes para encontrar la probabilidad condicional $P(w_i|x)$, la cual es la probabilidad condicional de la clase w_i dado un vector x [Duda73].

$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)} \tag{2.1}$$

donde $P(w_i)$ es conocida como la probabilidad a priori de que ocurra la clase w_i, esta probabilidad es estimada del ambiente, verificando el número de ocurrencias para cada una de las clases [Duda73]. Cuando no se cuenta con esta información del ambiente, lo mejor es asumir que la probabilidad de que ocurra cada clase w_i es la misma para todas, es decir, tienen una distribución uniforme, por lo tanto $P(w_i) = 1/C$.

$P(x|w_i)$ es conocida como la verosimilitud y expresa la probabilidad condicional de que un vector x se presente dado un objeto de la clase w_i, es decir, que tan creíble es que un objeto de la clase w_i tenga las características del vector x [Bishop06]. En la Figura 2.2 se muestra una gráfica donde se pueden observar 2 curvas donde cada una corresponde a una clase, estas curvas representan la densidad de probabilidad de medir un vector de características x unidimensional, dado que se sabe que pertenece a la clase w_i. Esto quiere decir que para cada valor que adopta x, cada curva expresa la densidad de probabilidad de que x se presente en la clase correspondiente a dicha curva. A esta probabilidad también se le conoce como probabilidad condicionada a la clase.

Por último, $P(x)$ puede ser visto como un factor de escala el cual garantiza que la suma de probabilidades $P(w_i|x)$ sea igual a 1, para $i = 0, 1, ..., C-1$. Lo que nos dice el teorema de Bayes es que al observar un vector x, podemos convertir una probabilidad a priori $P(w_i)$ en una probabilidad a posteriori $P(w_i|x)$, dada la medición del vector x.

De esta manera, podemos considerar que la clase a la que pertenece el vector x, es aquella para la cual $P(w_i|x)$ es mayor, es decir, elegimos la clase w_i si:

$$P(w_i|x) > P(w_j|x) \qquad \forall j \neq i \tag{2.2}$$

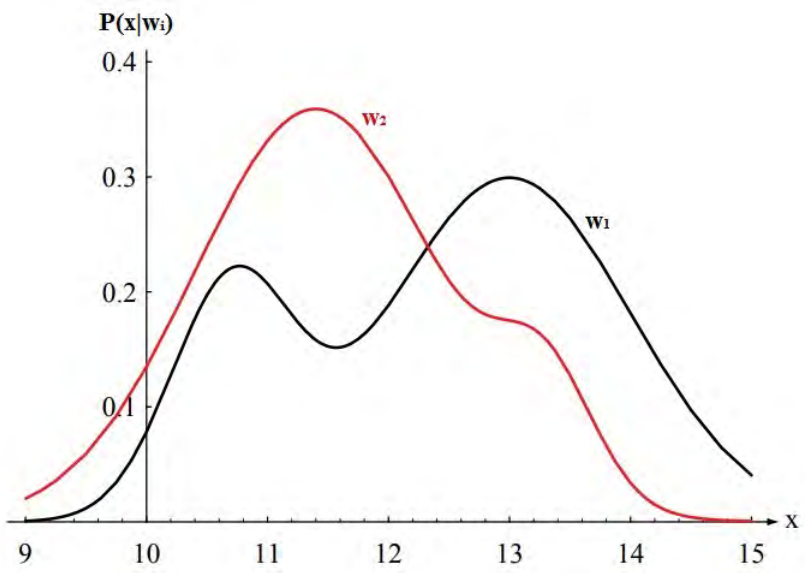


Figura 2.2: Ejemplo de verosimilitud para 2 clases [Duda73].

Christopher Bishop define que calcular $P(x|w_j)$ puede lograrse por medio de un conjunto de entrenamiento. Dado un conjunto de muestras $\{x_0, x_1, ..., x_N\}$ donde cada muestra es un vector de características que pertenece a una clase conocida w_j, es posible ajustar parámetros de un modelo adaptativo. En el caso de un segmentador bayesiano, los parámetros que usualmente se ajustan tienen que ver con distribuciones de probabilidad (por ejemplo, media y varianza). Posteriormente, estos parámetros son utilizados para calcular la probabilidad $P(x|w_j)$ para una clase en específico [Bishop06].

Si lo que queremos es segmentar una imagen, primero tenemos que representar dicha imagen como una matriz de vectores de características. Primero consideremos a una imagen como una matriz I de tamaño $n_r \times n_c$, cada pixel de la imagen será expresado como $I(n, m)$ el cual es un vector que se encuentra en la posición (n, m) de la matriz. Si la imagen contiene información a color, cada pixel será expresado como el vector $I(n, m) = [I_R(n, m), I_G(n, m), I_B(n, m)]$, donde cada elemento del vector representa una de las componentes de color de dicho pixel, donde $I_R(n, m)$ representa rojo, $I_G(n, m)$ representa verde e $I_B(n, m)$ representa el azul.

Para determinar a qué clase de color pertenece un pixel de la imagen dada, podemos

expresar cada pixel como el vector de características $x(n,m) = [I_R(n,m), I_G(n,m), I_B(n,m)]$ donde (n,m) es la posición del pixel en la imagen. Luego calculamos las probabilidades $P(x(n,m)|w_i)$, $P(w_i)$ y $P(x(n,m))$ para así poder calcular la probabilidad condicional $P(w_i|x(n,m))$ con (2.1); para determinar a que clase pertenece $x(n,m)$ aplicamos (2.2).

A continuación se presentan dos experimentos donde se puso a prueba la eficacia de aplicar un segmentador Bayesiano. En la Figura 2.3 se muestra una imagen artificial, la cual fue creada para aplicar una segmentación de 3 clases. La imagen fue construida en escala de grises y dividida en 3 secciones, donde cada sección representa una clase diferente, la sección de la izquierda representa la clase w_0, la sección central representa la clase w_1, y la sección de la derecha representa la clase w_2. Cada sección fue construida generando números aleatorios siguiendo una distribución de probabilidad normal, la cual esta dada por (2.3):

$$N(x(n,m)|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x(n,m)-\mu)^2}{2\sigma^2}} \tag{2.3}$$

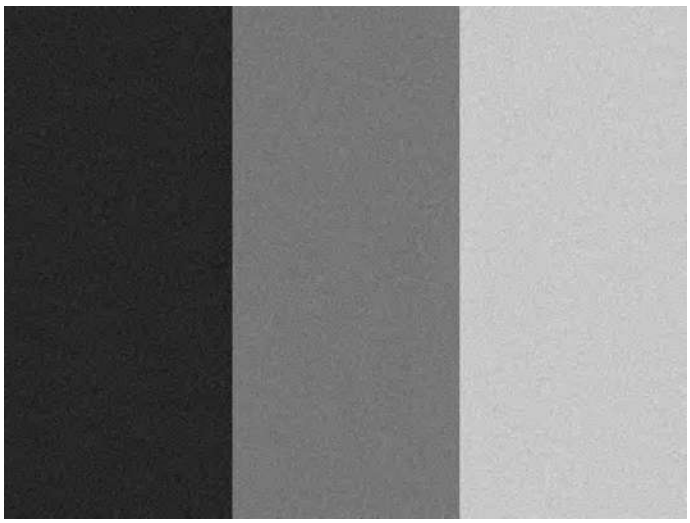


Figura 2.3: Imagen utilizada para el experimento 1. Una imagen artificial para 3 clases.

Para el primer experimento, la clase w_0 fue construida con una media $\mu_0 = 40$, la clase w_1 con $\mu_1 = 120$ y la clase w_2 con $\mu_2 = 200$, las 3 clases fueron construidas con una desviación estandar $\sigma_i = 8$, $i = 0, 1, 2$. En la Figura 2.4 se muestra el histograma correspondiente a este experimento, en el cual se puede observar la distribución de los datos

para las 3 clases.

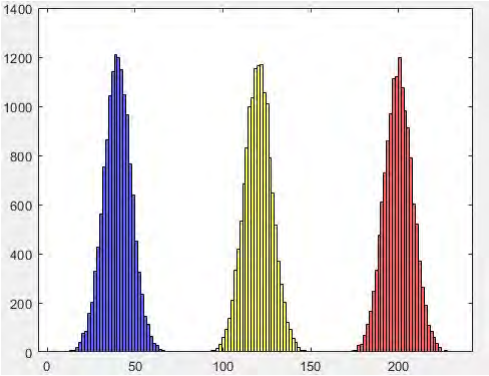


Figura 2.4: Histograma del experimento 1. El histograma azul corresponde a la clase w_0, el amarillo corresponde a la clase w_1, y el rojo corresponde a la clase w_2.

Para llevar a cabo la segmentación, primero expresamos $x(n, m)$ como un vector de características con información del color de un pixel de la imagen. Dado que la imagen está en escala de grises, el vector $x(n, m)$ solo contiene un valor el cual representa el tono de gris del pixel con rango entre 0 y 255. Después, se utiliza el teorema de Bayes aplicando (2.1). Dado que no tenemos información del ambiente para este experimento, asumimos que las probabilidades a priori de las clases $P(w_i)$ tienen una distribución uniforme, es decir, $P(w_i) = \frac{1}{3}$, $i = 0, 1, 2$.

Para calcular $P(x(n, m))$ aplicamos la regla de la probabilidad total, utilizando la regla de la probabilidad total, el cálculo de $P(x(n, m))$ para 3 clases está dado por:

$$P(x(n,m)) = \sum_{j=0}^{2} P(x(n,m)|w_j)P(w_j)$$

Por último, calculamos la probabilidad $P(x(n, m)|w_i)$. Para este experimento, sabemos que los patrones de cada clase siguen una distribución de probabilidad normal, dadas por una media y una varianza, como sabemos que estos parámetros son los que describen la misma clase, podemos expresar cada clase w_i como un conjunto de parámetros $\{\mu_i, \sigma_i\}$ y por lo tanto, la probabilidad condicional $P(x(n, m)|w_i)$ puede expresarse de la siguiente forma:

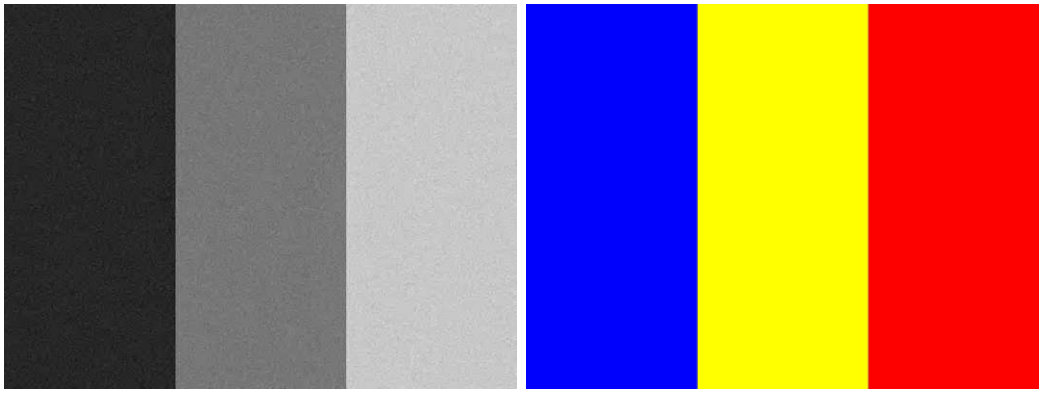
$$P(x(n,m)|w_i) = N(x(n,m)|\mu_i, \sigma_i)$$

Para este experimento los parámetros de probabilidad para cada clase μ_0, μ_1, μ_2 así como σ_0, σ_1, σ_2 ya se conocen, por lo tanto, no es necesario aplicar un entrenamiento. La ecuación resultante para calcular $P(w_i|x(n,m))$ se muestra a continuación.

$$P(w_i|x(n,m)) = \frac{N(x(n,m)|\mu_i,\sigma_i)(\frac{1}{3})}{\sum_{j=0}^{2}\left[N(x(n,m)|\mu_j,\sigma_j)\frac{1}{3}\right]} = \frac{N(x(n,m)|\mu_i,\sigma_i)}{\sum_{j=0}^{2}N(x(n,m)|\mu_j,\sigma_j)} = K{\cdot}N(x(n,m)|\mu_i,\sigma_i) \tag{2.4}$$

donde

$$K = \frac{1}{\sum_{j=0}^{2}N(x(n,m)|\mu_j,\sigma_j)}$$



(a) **(b)**

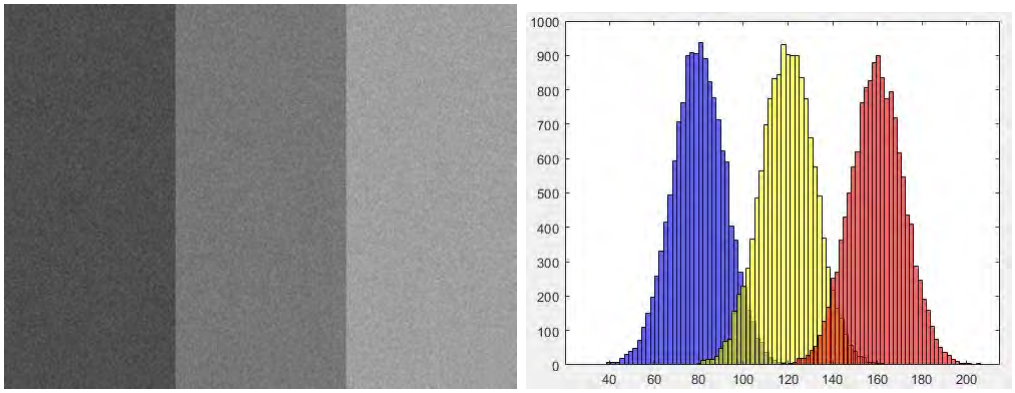
Figura 2.5: Resultados de segmentación para el experimento 1. En (a) se muestra la imagen original, en (b) se muestra el resultado de la segmentación, la clase w_0 se muestra en color azul, la clase w_1 en amarillo y la clase w_2 en rojo.

Los resultados de esta segmentación son mostrados en la Figura 2.5, en (a) se muestra la imagen original a segmentar, en (b) se muestra la imagen resultante de la segmentación, donde la clase w_0 es representada con color azul, la clase w_1 con amarillo y la clase w_2 con rojo. Para verificar la precisión de la segmentación se optó por utilizar los coeficientes de Tanimoto [Tanimoto05], este coeficiente permite conocer la similitud entre 2 conjuntos y está dado por la siguiente expresión:

$$T(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.5}$$

donde A y B son dos conjuntos, A corresponde al conjunto de pixeles que integran una clase, y B representa el conjunto de pixeles que fueron correctamente clasificados para esta misma clase.

La función (2.5) devuelve un valor que se encuentra entre 0 y 1, donde 1 representa una segmentación perfecta, si multiplicamos este valor por 100, tendremos un valor en porcentaje. Para el primer experimento, se tuvo una precisión del 100 % para la segmentación de la clase w_0, w_1 y w_2 respectivamente, por lo que se puede considerar como una segmentación exitosa.



(a) **(b)**

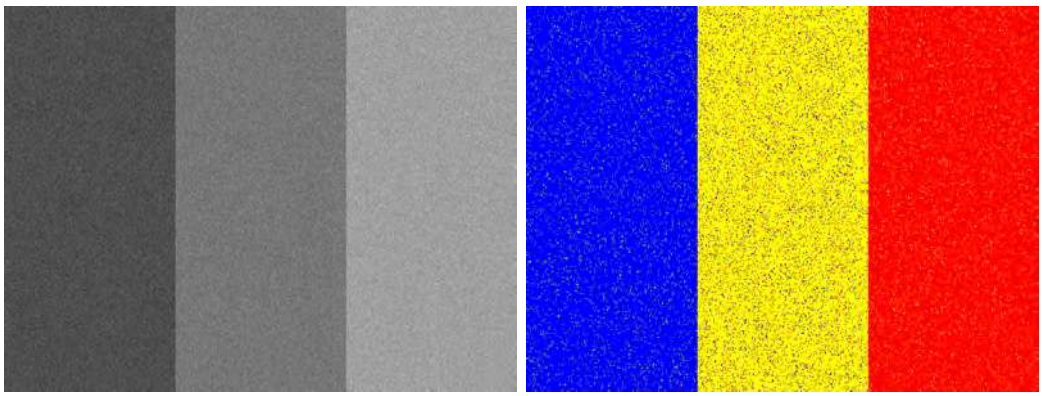
Figura 2.6: Imagen utilizada para el experimento 2. En (a) se muestra la imagen original, en (b) se muestra el histograma para dicha imagen.

Para el segundo experimento, también se construyó una imagen artificial, siguiendo las mismas condiciones que el experimento 1, con la diferencia de que para este caso, la clase w_0 fue generada con una media $\mu_1 = 80$, la clase w_1 con $\mu_2 = 120$ y la clase w_2 con $\mu_3 = 160$, las 3 clases fueron creadas con una desviación estandar $\sigma = 12$. En la Figura 2.6 (a) se muestra la imagen utilizada en este experimento, en (b) se muestra su histograma correspondiente.

Como se puede observar en este histograma, la distribución de los datos para cada clase se encuentra más cerca de las demás clases, lo que provoca que los histogramas se sobrepongan, esto ocurre a pesar de que su media sigue estando separada.

Para este experimento, el método de segmentación aplicado fue el mismo que para

el caso del experimento 1, los resultados de esta segmentación son mostrados en la Figura 2.7, en (a) se muestra la imagen original, en (b) se muestra la imagen resultante de la segmentación. Se puede observar a simple vista que la eficiencia de esta segmentación es menor que para el experimento 1, se puede distinguir que en la sección correspondiente a la clase w_0 varios pixeles fueron clasificados como clase w_1, en la sección correspondiente a la clase w_1, varios pixeles fueron clasificados como clase w_0 y w_2, y en la sección correspondiente a la clase w_2, varios pixeles fueron clasificados como clase w_1. Esto ocurre porque, como se mostró en el histograma de la Figura 2.6 (b) la distribución de probabilidad de la clase w_0 está muy cerca de la clase w_1, y la clase w_1 está muy cerca de la clase w_2, la razón por la que los pixeles correspondientes a la clase w_0 no son clasificados como clase w_2 y viceversa, es porque sus distribuciones de probabilidad se encuentran lo suficientemente alejadas para que la probabilidad de que una de ellas sea erróneamente clasificadas es cercana a 0.



(a) **(b)**

Figura 2.7: Resultados de segmentación para el experimento 2. En (a) se muestra la imagen original, en (b) se muestra el resultado de la segmentación, la clase w_0 se muestra en color azul, la clase w_1 en amarillo y la clase w_2 en rojo.

El porcentaje de precisión de esta segmentación es de un 91.15 % para la clase w_0, 82.13 % para la clase w_1 y 90.67 % para la clase w_2. La precisión total se calcula aplicando un promedio de estos porcentajes, para este experimento la precisión total es de un 87.98 %. El motivo de que esta segmentación sea menos precisa, es debido a la cercanía que hay entre las densidades de probabilidad. Por ejemplo, si la densidad de probabilidad de un vector

$x(n, m)$ que pertenece a la clase w_0 se encuentra lo suficientemente lejos de su media, pero sí lo suficientemente cerca de la media de la clase w_1, es más probable que este sea clasificado en la clase w_1, siendo ésta una clasificación errónea.

Por último, se realizaron 8 experimentos más con este tipo de segmentación, los resultados de estos experimentos (incluyendo los 2 anteriores) son mostrados en la Tabla 2.1.

Experimento	μ_0, μ_1, μ_2	σ	Porcentaje
1	25,125,225	8	100 %
2	25,125,225	16	99.86 %
3	25,125,225	24	97.37 %
4	40,120,200	8	100 %
5	40,120,200	16	99 %
6	40,120,200	24	93.32 %
7	80,120,160	8	98.96 %
8	80,120,160	12	87.98 %
9	80,120,160	16	85.07 %
10	80,120,160	24	72.22 %

Tabla 2.1: Resultados de la segmentación utilizando (2.4).

2.3. Conclusiones

Con estos resultados se puede concluir que un clasificador Bayesiano es útil para llevar a cabo tareas de segmentación de imágenes. Sin embargo, pueden presentarse situaciones en las que esta clasificación sea imprecisa o incluso falle. En algunos problemas del mundo real, tener un buen segmentador es un factor importante y decisivo en la solución del problema, por lo que es necesario proponer mejoras para este tipo de clasificadores. En el capítulo 3 se presenta la coherencia espacial, el cual es un método que consiste en forzar los pixeles que se encuentran dentro de una vecindad a ser clasificados para la misma clase,

por lo que se puede resolver el problema mostrado en la Figura 2.7.

Capítulo 3

Segmentación por Maximización de Funciones Discriminantes

En este capítulo se presenta una extensión del algoritmo de Segmentación por Maximización de Funciones Discriminantes (SMDF), método que fue propuesto por *Calderón et al.* [Calderon15]. El algoritmo extendido fue desarrollado para clasificar varias clases, mientras que el método SMDF sólo opera con dos clases. La segmentación se lleva a cabo en centésimas de segundos.

3.1. Introducción

El método SMDF consiste en llevar a cabo una segmentación binaria por maximización de funciones discriminantes, empleando el histograma para calcular la probabilidad de que el vector x pertenezca a la clase w_i. Estas probabilidades son expresadas en términos de funciones discriminantes de manera que se tienen dos funciones discriminantes (una por clase). Para determinar a que clase pertenece x, se propone una función que consiste en una combinación lineal de ambas funciones discriminantes, después se propone un método de optimización para encontrar el máximo de dicha función. Para construir el histograma se utiliza un conjunto de entrenamiento el cual contiene valores dados por pixeles que pertenecen a la clase w_i, teniendo así un histograma para cada clase. Para mejorar la segmentación,

el método utiliza coherencia espacial la cual consiste en forzar los pixeles que se encuentran dentro de una vecindad a ser clasificados para la misma clase [Calderon15]. En este capítulo se presenta una extensión del método SMDF, con el propósito de que este pueda ser utilizado para segmentar más de dos clases.

Primero consideremos que tenemos una imagen a color I dada por una matriz, definimos el vector $I(n,m)$ como un pixel de la imagen I que se encuentra en el renglón n y la columna m. El problema consiste en asignar cada pixel $I(n,m)$ a una etiqueta $q(n,m)$ la cual representa la clase de color a la cual pertenece el pixel $I(n,m)$. Por lo tanto, definimos $q(n,m)$ como una variable discreta cuyos valores se encuentran dentro del conjunto $\Phi = \{0,1,...,C-1\}$, donde C es el número de clases posibles.

Dado que I corresponde a una imagen a color, cada pixel $I(n,m)$ está dado por un vector con información de su color, por lo que definimos $x(n,m)$ como un vector de características para el pixel $I(n,m)$ de la forma $x(n,m) = [I_R(n,m), I_G(n,m), I_B(n,m)]$, donde $I_R(n,m), I_G(n,m), I_B(n,m)$ representan los valores de color R,G,B para el pixel $I(n,m)$. De esta forma, podemos expresar la probabilidad condicional como $P(w_i|x(n,m))$. Esta probabilidad puede ser escrita en términos de funciones discriminantes [Duda73]. Las funciones discriminantes son funciones utilizadas en los problemas de clasificación donde para el vector dado $x(n,m)$, se escoge la clase para la cual la función discriminante es mayor. Para expresar $P(w_i|x(n,m))$ en estos términos, definiremos una función discriminante $f_i(x(n,m))$ correspondiente a la i-ésima clase de la siguiente forma:

$$f_i(x(n,m)) = P(w_i|x(n,m)) \tag{3.1}$$

donde $f_i(x(n,m))$ es la función discriminante para la clase w_i, ésta función recibe un vector de características $x(n,m)$ de 3 dimensiones y devuelve un valor de probabilidad, por lo que también se expresa como $f_i : \mathbb{R}^3 \to \mathbb{R}$. De forma similar a (2.2), queremos asignar la etiqueta $q(n,m)$ al vector de características $x(n,m)$ para el cual $f_i(x(n,m))$ es mayor, teniendo como resultado la siguiente regla:

$$q(n,m) = i \leftrightarrow \exists_{i\in\Phi}\forall_{j\in\Phi}[(i \neq j) \wedge f_i(x(n,m)) > f_j(x(n,m))] \tag{3.2}$$

Por otro lado, la regla (3.2) puede ser expresada como un problema de optimización, para lograr esto, definimos una variable continua $\hat{q}(n,m)$ cuyo valor se encuentra

en el intervalo $[0, C-1]$, de esta manera, nuestro problema de optimización consiste en que para una función $F(\hat{q}(n,m))$ queremos encontrar el valor $\hat{q}(n,m)$ que la maximize, para que se cumpla la regla (3.2), es necesario que esta función interpole por los puntos $(0, f_0(x(n,m)))$,$(1, f_1(x(n,m)))$,...,$(C-1, f_{C-1}(x(n,m)))$, ya que queremos que cuando $\hat{q}(n,m) = i$, tengamos el valor $F(\hat{q}(n,m)) = f_i(x(n,m))$. Esto se puede lograr si definimos a $F(\hat{q}(n,m))$ como una función lineal a trozos, la cual se explica en la siguiente sección.

3.2. Cálculo de la segmentación a partir de una función lineal a trozos

Una función lineal a trozos consiste en una función que corresponde a un conjunto de funciones afines, donde para nuestro caso, queremos que cada función afín interpole entre 2 puntos $(k, f_k(x(n,m)))$, $(k+1, f_{k+1}(x(n,m)))$ para un conjunto de puntos dados. Para que una función afín interpole en estos puntos, podemos expresar cada función como una combinación lineal, de tal manera que expresemos una función lineal a trozos de la siguiente forma:

$$F(\hat{q}(n,m)) = \begin{cases} \hat{q}(n,m)(f_1(x(n,m)) - f_0(x(n,m))) & 0 \leq \hat{q}(n,m) < 1 \\ \quad +f_0(x(n,m)) & \\ \hat{q}(n,m)(f_2(x(n,m)) - f_1(x(n,m))) & 1 \leq \hat{q}(n,m) < 2 \\ \quad +2f_1(x(n,m)) - f_2(x(n,m)) & \\ \hat{q}(n,m)(f_3(x(n,m)) - f_2(x(n,m))) & 2 \leq \hat{q}(n,m) < 3 \\ \quad +3f_2(x(n,m)) - 2f_3(x(n,m)) & \\ \quad \vdots & \quad \vdots \\ \hat{q}(n,m)(f_{C-1}(x(n,m)) - f_{C-2}(x(n,m))) & C-2 \leq \hat{q}(n,m) \\ +(C-1)f_{C-2}(x(n,m)) - (C-2)f_{C-1}(x(n,m)) & \hat{q}(n,m) \leq C-1 \end{cases} \tag{3.3}$$

Se puede observar que para el caso en que $\hat{q}(n,m) = k$, entonces $F(\hat{q}(n,m)) = f_k(x(n,m))$. En la Figura 3.1 se muestra un ejemplo de esta función. El objetivo es asignar el valor $\hat{q}(n,m)$ que cumpla la regla (3.2). Podemos expresar esto como un problema de optimización, donde queremos encontrar el valor de $\hat{q}^*(n,m)$ que maximice la función $F(\hat{q}(n,m))$.

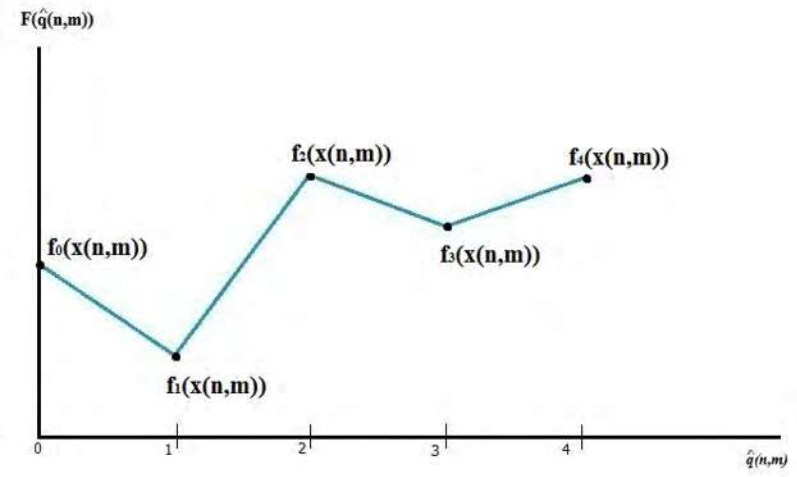


Figura 3.1: Función lineal a trozos para 5 clases.

Primero encontramos el máximo para cada función afín de manera individual y posteriormente, encontramos el máximo de todas estas funciones. Para el caso de una función afín con la restricción $k \leq \hat{q}(n,m) \leq k+1$, el máximo siempre se encuentra ya sea en k o $k+1$, al evaluar estos valores en la función $F(\hat{q}(n,m))$ dada, el problema de maximización puede ser expresado como:

$$\text{máx}\, F(\hat{q}(n,m)) = \text{máx} \begin{cases} \text{máx}[f_0(x(n,m)), f_1(x(n,m))] & 0 \leq \hat{q}(n,m) < 1 \\ \text{máx}[f_1(x(n,m)), f_2(x(n,m))] & 1 \leq \hat{q}(n,m) < 2 \\ \text{máx}[f_2(x(n,m)), f_3(x(n,m))] & 2 \leq \hat{q}(n,m) < 3 \\ \vdots & \vdots \\ \text{máx}[f_{C-2}(x(n,m)), f_{C-1}(x(n,m))] & C-2 \leq \hat{q}(n,m) \leq C-1 \end{cases} \tag{3.4}$$

Por lo que de forma equivalente, el problema puede ser resuelto con (3.5):

$$\begin{aligned} \text{máx}\, F(\hat{q}(n,m)) &= \text{máx}[f_0(x(n,m)), f_1(x(n,m)), ..., f_{C-1}(x(n,m))] \\ \hat{q}^*(n,m) &= \arg\max[F(0), F(1), ..., F(C-1)] \end{aligned} \tag{3.5}$$

3.3. Coherencia espacial

Cuando un objeto presente en una imagen que pertenece a una clase dada es clasificado, se espera que todos los pixeles pertenecientes a este objeto pertenezcan a la misma clase. Sin embargo, como se observó en la Figura 2.7 (b), hay ocasiones donde algunos modelos de segmentación no pueden clasificar correctamente algunos pixeles de la imagen, debido a la presencia de ruido o que las distribuciones de probabilidad de cada clase cambien debido a alteraciones en el ambiente (como cambios de iluminación o presencia de sombras, entre otros). Este problema provoca que pixeles vecinos que pertenecen al mismo objeto son clasificados para diferentes clases cuando estos en realidad debían ser clasificados para el mismo objeto.

La coherencia espacial consiste en restringir las áreas pertenecientes a un objeto, forzando a los pixeles que se encuentran dentro de una vecindad a ser clasificados para la misma clase. Podemos proponer una función que mide la diferencia entre una etiqueta $\hat{q}(n,m)$ y 3 de sus vecinos dados por el conjunto $V(n,m) = \{\hat{q}(n-1,m), \hat{q}(n,m-1), \hat{q}(n-1,m-1)\}$, estas diferencias deben ser menor que un valor ϵ por lo que se expresan como las restricciones dadas por (3.6):

$$\begin{aligned} |\hat{q}(n,m) - \hat{q}(n-1,m)| &< \epsilon \\ |\hat{q}(n,m) - \hat{q}(n,m-1)| &< \epsilon \\ |\hat{q}(n,m) - \hat{q}(n-1,m-1)| &< \epsilon \end{aligned} \tag{3.6}$$

Es deseable que estas diferencias tengan un valor pequeño positivo ϵ, mientras más pequeño sea el valor de ϵ más aproximada es $\hat{q}(n,m)$ con sus vecinos. Para expresar estas restricciones sin valor absoluto podemos escribirlas como (3.7):

$$\begin{aligned} \hat{q}(n,m) - \hat{q}(n-1,m) &< \epsilon \\ \hat{q}(n-1,m) - \hat{q}(n,m) &< \epsilon \\ \hat{q}(n,m) - \hat{q}(n,m-1) &< \epsilon \\ \hat{q}(n,m-1) - \hat{q}(n,m) &< \epsilon \\ \hat{q}(n,m) - \hat{q}(n-1,m-1) &< \epsilon \\ \hat{q}(n-1,m-1) - \hat{q}(n,m) &< \epsilon \end{aligned} \tag{3.7}$$

De esta manera, podemos expresar la segmentación como una función $F_s(\hat{q})$ correspondiente al promedio de todas las funciones discriminantes resultantes de evaluar $\hat{q}(n,m)$ en

$F(\hat{q}(n,m))$ para toda la imagen. Esto a su vez se expresa como un problema de optimización donde queremos encontrar el promedio máximo para esta función. El problema de maximización es mostrado a continuación.

$$\begin{gathered}
\text{máx}\, F_s(\hat{q}) = \text{máx}\, \tfrac{1}{n_r \times n_c} \sum_{n=0}^{nr-1} \sum_{m=0}^{nc-1} F(\hat{q}(n,m)) \\
sujeto \quad a \\
\hat{q}(n,m) - \hat{q}(n-1,m) < \epsilon \\
\hat{q}(n-1,m) - \hat{q}(n,m) < \epsilon \\
\hat{q}(n,m) - \hat{q}(n,m-1) < \epsilon \\
\hat{q}(n,m-1) - \hat{q}(n,m) < \epsilon \\
\hat{q}(n,m) - \hat{q}(n-1,m-1) < \epsilon \\
\hat{q}(n-1,m-1) - \hat{q}(n,m) < \epsilon
\end{gathered} \tag{3.8}$$

Dado que el valor $\frac{1}{n_r \times n_c}$ es una constante que no afecta la ubicación del máximo, ésta será omitida. Calcular el máximo de esta función puede ser logrado utilizando el método simplex, sin embargo, la complejidad de resolver el problema con el método simplex es de $O(2^{N_v})$ donde N_v es el número de vértices en la región de factibilidad [Calderon15]. Por otro lado, el método simplex solo puede ser utilizado en funciones convexas, lo cual no siempre ocurre en nuestro caso. Por este motivo, el método simplex no se considera adecuado para este problema.

Otra forma es dividir la imagen en un conjunto de ventanas de un tamaño $(2d+1) \times (2d+1)$ donde d es un número entero positivo, después resolvemos el problema de maximización dentro de cada ventana. De esta manera, el problema puede ser escrito como:

$$\begin{gathered}
\text{máx}\, F_s(\hat{q}) = \text{máx} \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} F(\hat{q}(k,l)) \\
sujeto \quad a \\
\hat{q}(k,l) - \hat{q}(k-1,l) < \epsilon \\
\hat{q}(k-1,l) - \hat{q}(k,l) < \epsilon \\
\hat{q}(k,l) - \hat{q}(k,l-1) < \epsilon \\
\hat{q}(k,l-1) - \hat{q}(k,l) < \epsilon \\
\hat{q}(k,l) - \hat{q}(k-1,l-1) < \epsilon \\
\hat{q}(k-1,l-1) - \hat{q}(k,l) < \epsilon
\end{gathered} \tag{3.9}$$

Se puede observar que si maximizamos $F(\hat{q})$ dentro de una ventana lo suficientemente pequeña y con un valor ϵ bajo, la solución $\hat{q}$ tiene poca variación, por lo que

podemos asumir que tiene un valor constante y la definimos como $\bar{q}_{n,m}$. De esta manera, las restricciones de coherencia espacial pueden ser escritas como:

$$\bar{q}_{n,m} \equiv \hat{q}(k,l) \tag{3.10}$$

Puesto que se cumple que:

$$\hat{q}(k,l) \approx \hat{q}(k-1,l) \approx \hat{q}(k,l-1) \approx \hat{q}(k-1,l-1)$$

Por lo tanto, maximizar (3.9) puede ser escrita asumiendo que la etiqueta para cada pixel es una constante $\bar{q}_{n,m}$, permitiéndonos omitir las restricciones.

$$\text{máx}\, F_s(\bar{q}_{n,m}) = \text{máx} \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} F(\bar{q}_{n,m})$$

Para simplificar el problema, podemos expresar la función lineal a trozos de la siguiente forma:

$$\text{máx}\, F_S(\bar{q}_{n,m}) =$$

$$\text{máx} \begin{cases} \text{máx}[\bar{q}_{n,m}(S_1(n,m) - S_0(n,m)) & 0 \leq \bar{q}_{n,m} < 1 \\ \quad +S_0(n,m)] & \\ \text{máx}[\bar{q}_{n,m}(S_2(n,m) - S_1(n,m)) & 1 \leq \bar{q}_{n,m} < 2 \\ \quad +2S_1(n,m) - S_2(n,m)] & \\ \text{máx}[\bar{q}_{n,m}(S_3(n,m) - S_2(n,m)) & 2 \leq \bar{q}_{n,m} < 3 \\ \quad +3S_2(n,m) - 2S_3(n,m)] & \\ \quad \vdots & \quad \vdots \\ \text{máx}[\bar{q}_{n,m}(S_{C-1}(n,m) - S_{C-2}(n,m)) & C-2 \leq \bar{q}_{n,m} \leq C-1 \\ +(C-1)S_{C-2}(n,m) - (C-2)S_{C-1}(n,m)] & \end{cases} \tag{3.11}$$

donde $S_i(n,m)$ corresponde a la suma de las funciones discriminantes dentro de una ventana para la clase i. $S_i(n,m)$ es calculada con (3.12).

$$S_i(n,m) = \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} f_i(x(k,l)) \tag{3.12}$$

La función dada por (3.11) puede ser resuelta de la misma forma que (3.4), donde para este caso, dicha función interpola en el conjunto de puntos $(0, S_0(n,m)), (1, S_1(n,m)), ...,$

$(C-1, S_{C-1}(n,m))$. Por lo tanto, para encontrar el máximo de $F_S(\bar{q}_{n,m})$, primero encontramos el máximo para cada combinación lineal y posteriormente, encontramos el máximo de todas las combinaciones lineales. Esto se logra aplicando la siguiente ecuación:

$$\max F_S(\bar{q}_{n,m}) = \max \begin{cases} \max[S_0(n,m), S_1(n,m)] & 0 \leq \bar{q}_{n,m} < 1 \\ \max[S_1(n,m), S_2(n,m)] & 1 \leq \bar{q}_{n,m} < 2 \\ \max[S_2(n,m), S_3(n,m)] & 2 \leq \bar{q}_{n,m} < 3 \\ \vdots & \vdots \\ \max[S_{C-2}(n,m), S_{C-1}(n,m)] & C-2 \leq \bar{q}_{n,m} \leq C-1 \end{cases} \tag{3.13}$$

De forma equivalente, el problema puede ser resuelto de la siguiente forma:

$$\begin{aligned} \max F_S(\bar{q}_{n,m}) &= \max[S_0(n,m), S_1(n,m), ..., S_{C-1}(n,m)] \\ \bar{q}^*_{n,m} &= \arg\max[F_S(0), F_S(1), ..., F_S(C-1)] \end{aligned} \tag{3.14}$$

La función $F_S(\bar{q}_{n,m})$ fue desarrollada haciendo una extensión de la misma función propuesta por *Calderón et al.* [Calderon15]. Para demostrar la equivalencia con esta función, se puede mostrar el caso de presentar la función $F_S(\bar{q}_{n,m})$ para 2 clases, en este caso dado que solo se necesita interpolar 2 puntos, $F_S(\bar{q}_{n,m})$ puede ser expresada con una sola función afín de la siguiente forma:

$$F_S(\bar{q}_{n,m}) = \bar{q}_{n,m}(S_1(n,m) - S_0(n,m)) + S_0(n,m)$$

Dado que $S_0(n,m)$ es un valor constante que no afecta el resultado de la segmentación, este puede ser omitido.

$$F_S(\bar{q}_{n,m}) = \bar{q}_{n,m}(S_1(n,m) - S_0(n,m))$$

Por otro lado, podemos expresar una función $S(n,m)$ de la siguiente forma:

$$\begin{aligned} S(n,m) &= S_1(n,m) - S_0(n,m) \\ &= \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} f_1(x(k,l)) - \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} f_0(x(k,l)) \\ &= \sum_{k=n-d}^{n+d} \sum_{l=m-d}^{m+d} \Delta f(k,l) \end{aligned}$$

donde

$$\Delta f(k,l) = f_1(x(k,l)) - f_0(x(k,l))$$

Como resultado tenemos la función:

$$F_S(\bar{q}_{n,m}) = \bar{q}_{n,m} S(n,m)$$

La cual es equivalente a la función presentada en el artículo previamente mencionado.

Llevar a cabo el cálculo de las funciones $S_i(n,m)$ requiere procesar los datos de toda una ventana de tamaño $(2d+1)\times(2d+1)$, este cálculo debe realizarse para cada uno de los pixeles de la imagen por lo que computar S_i requiere de un tiempo $Ts(A) = O(A\times(2d+1)^2)$ donde A es el área de la imagen dada por $A = n_r \times n_c$. Considerando que esto debe hacerse para cada clase del clasificador, es necesario proponer métodos que mejoren esta complejidad.

Una propuesta puede ser realizar el cálculo de imágenes incrementales, método que fue propuesto por Viola y Jones en su artículo [Viola01], este consiste en proponer una función $G_i(K,L)$ la cual es una suma de todas las funciones discriminantes desde los índices $(0,0)$ de la imagen hasta una posición (K,L), para $K = 0,1,...,n_r$ y $L = 0,1,...,n_c$.

$$G_i(K,L) = \sum_{k=0}^{K}\sum_{l=0}^{L} f_i(x(k,l)) \tag{3.15}$$

Por lo tanto, $S_i(n,m)$ puede ser calculada de la siguiente forma:

$$\begin{aligned} S_i(n,m) = {} & G_i(n+d, m+d) \\ & -G_i(n+d, m-d-1) \\ & -G_i(n-d-1, m+d) \\ & +G_i(n-d-1, m-d-1) \end{aligned} \tag{3.16}$$

Llevar a cabo el cálculo de $G_i(n,m)$ puede ser logrado recorriendo los pixeles de la imagen una sola vez, por lo que requiere de un tiempo $Ts(A) = O(A)$. Dado que $G_i(n,m)$ solo se debe calcular una vez, y $S_i(n,m)$ puede ser calculado con la suma de 4 valores, la complejidad del algoritmo se reduce considerablemente. El método para llevar a cabo la segmentación con coherencia espacial es mostrado en el **Algoritmo 1**. Este método fue creado a partir de hacer una extensión del algoritmo de Segmentación por

Maximización de Funciones Discriminantes (SMDF), el cual fue propuesto por *Calderón et al* en su artículo [Calderon15], por lo que este será nombrado SMDF extendido.

Algoritmo 1: SMDF extendido.

Datos: Una imagen I

Resultado: $\bar{q}_{n,m}$ para cada pixel de I

Computar $f_0, f_1, ..., f_{C-1}$ para cada pixel de I

Computar $G_0, G_1, ..., G_{C-1}$ con (3.15) para cada pixel de I

para $n = 0$ ***hasta*** $n_r - 1$ **hacer**

  para $m = 0$ ***hasta*** $n_c - 1$ **hacer**

    Computar $S_0, S_1, ..., S_{C-1}$ con (3.16)

    Computar $\bar{q}_{n,m}$ *para* $F_S(\bar{q}_{n,m})$ con (3.14)

  fin

fin

devolver $\bar{q}_{n,m}$ para cada pixel de I

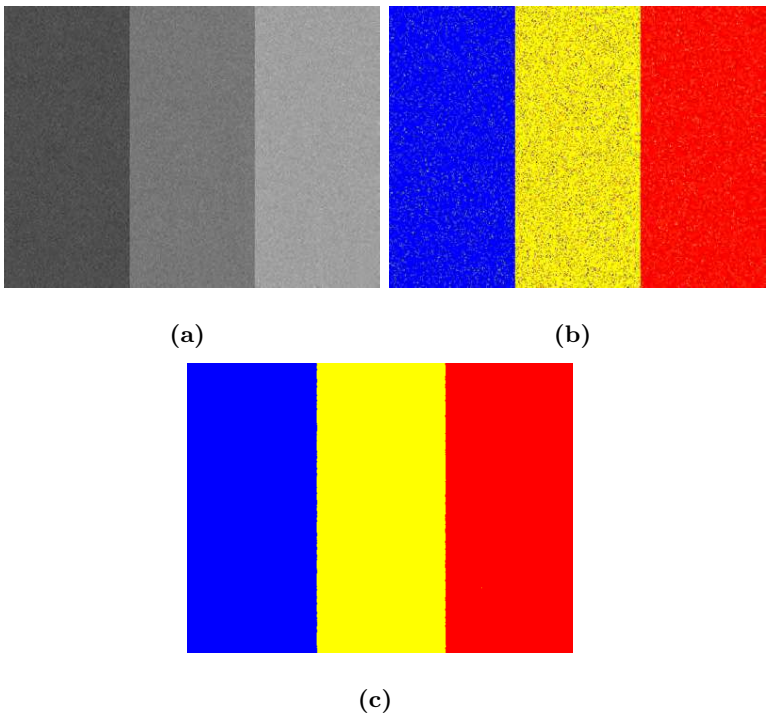


Figura 3.2: Segmentación con coherencia espacial para 3 clases. En (a) se muestra la imagen original, En (b) se muestra la imagen segmentada sin coherencia espacial, en (c) se muestra la imagen segmentada utilizando coherencia espacial.

En la Figura 3.2 se muestran los resultados de aplicar la segmentación con coherencia espacial para el experimento de la Figura 2.7, en (a) se muestra la imagen original, en (b) se muestra la segmentación sin coherencia espacial y en (c) se muestra la segmentación con coherencia espacial, la cual fue llevada a cabo con $d = 1$. Utilizando los coeficientes de Tanimoto, la precisión de la segmentación con coherencia espacial es de 98.64 %, mientras que sin coherencia espacial esta precisión es de 87.98 %.

3.4. Entrenamiento

Existen casos en los que es necesario llevar a cabo múltiples experimentos en varios ambientes con diferente iluminación, si esto ocurre es necesario realizar un nuevo entrenamiento para cada ambiente ya que la distribución de color de los objetos presentes en el ambiente puede variar.

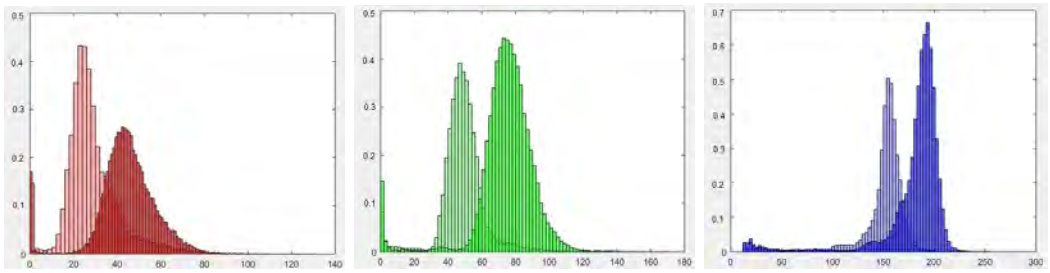
Si queremos llevar a cabo una segmentación tolerante ante los cambios de iluminación, es necesario llevar a cabo el entrenamiento con múltiples imágenes donde la iluminación es diferente para cada imagen. Considérese que queremos segmentar las imágenes mostradas en la Figura 3.3, tanto (a) como (b). En ambas imágenes queremos separar los objetos de color azul del resto de la imagen, por lo que para este caso definimos 2 clases, la clase azul y la clase fondo.

(a) **(b)**

Figura 3.3: Ejemplo de 2 imágenes con cambios de iluminación.

En la Figura 3.4 se muestran los histogramas correspondientes a la clase azul

para ambas imágenes, en (a) se muestran los 2 histogramas para el canal rojo para ambas imágenes, en (b) se muestran los 2 histogramas para el canal verde y en (c) se muestran los 2 histogramas para el canal azul. Se puede observar en esta figura que la distribución de los datos es diferente para cada imagen.



(a) **(b)** **(c)**

Figura 3.4: Histogramas para la clase Azul para la figura 3.3. en (a) se muestra el histograma correspondiente al canal rojo, en (b) al canal verde y en (c) al canal azul.

En el mejor de los casos, la unión de ambos histogramas para cada canal forman una distribución que puede ser aproximada con una distribución compuesta por una mezcla de Gaussianas. Si queremos encontrar los parámetros que la conforman, se pueden implementar métodos de estimación de parámetros como Expectation Maximization (EM) [Dempster77]. Sin embargo, optar por utilizar EM puede no ser una buena idea para nuestro propósito, ya que este método puede tener varias desventajas que dificulten cumplir con los objetivos de esta tesis. Estas desventajas pueden ser las siguientes:

- La convergencia del algoritmo EM es bastante lenta. El entrenamiento rápido es una prioridad, ya que cuando existen cambios de iluminación en el ambiente, es necesario volver a entrenar el modelo para tener una segmentación con buena precisión. Por este motivo, el algoritmo EM puede no ser apropiado.

- Los cambios de iluminación pueden modificar las distribuciones de los datos como ya se mostró en la Figura 1.1 y 3.4. Para proponer un entrenamiento robusto a los cambios de iluminación, se puede entrenar considerando también aquellos datos sometidos a los cambios de iluminación, sin embargo, esto nos puede llevar a distribuciones de

probabilidad cada vez más complejas y difíciles de modelar matemáticamente.

Una mejor propuesta puede ser el uso del histograma, el cual fue introducido por Karl Pearson [Pearson94]. El histograma es la distribución de probabilidad empírica para una variable discreta. Para construir un histograma con un conjunto de entrenamiento dado $x = \{x_0, x_1, x_2, ..., x_{N-1}\}$, utilizaremos la definición de histograma dada por Prince [Prince12], la cual consiste en la siguiente expresión:

$$h[b] = \sum_{i=0}^{N-1} \delta(x_i - b) \tag{3.17}$$

donde $\delta(z)$ es la función delta que está dada por:

$$\delta(z) = \begin{cases} 1 & si \quad z = \mathbf{0} \\ 0 & en \quad caso \quad contrario \end{cases}$$

El histograma recibe un vector b el cual para el caso de imágenes a color está expresado como $b = [b_R, b_G, b_B]$. La función $\delta(z)$ recibe un vector z el cual devuelve 1 si z es igual al vector cero y 0 en caso contrario. Para el caso de una imagen a color, se tiene un histograma de 3 dimensiones.

El uso del histograma es preferible porque el cálculo del mismo es bastante rápido, ya que este puede ser calculado recorriendo los elementos del conjunto de entrenamiento una sola vez, además, el histograma puede modelar distribuciones de probabilidad que en ocasiones no pueden ser modeladas de forma matemática.

Para este caso, el entrenamiento será llevado a cabo con un conjunto de entrenamiento $x = \{x_0, x_1, ..., x_N\}$ donde x_i es un vector correspondiente a un pixel que pertenece a una clase de color, cada elemento x_i puede ser expresado por un vector dado por $[x_{i_R}, x_{i_G}, x_{i_B}]$, entonces, se aplica la función del histograma mostrada en (3.17) la cual corresponde a un histograma de 3 dimensiones. De esta manera la probabilidad $P(x(n,m)|w_i)$ puede ser expresada como:

$$P(x(n,m)|w_i) = \frac{1}{H_i} h_i[x(n,m)] \tag{3.18}$$

donde h_i es el histograma para la clase w_i con el vector de características $x(n,m)$, $\frac{1}{H_i}$ es una constante de normalización que garantiza que la suma de todos los valores del

histograma h_i sea igual a 1. Aplicando el teorema de Bayes dado por (2.1), tenemos lo siguiente:

$$P(w_i|x(n,m)) = \frac{\frac{1}{H_i}h_i[x(n,m)]P(w_i)}{P(x(n,m))}$$

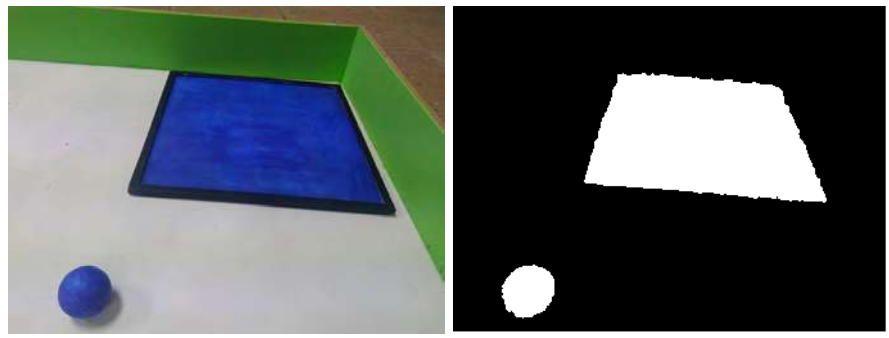
donde la probabilidad $P(w_i)$ será expresada como una probabilidad uniforme $P(w_i) = 1/C$ ya que no se cuenta con suficiente información del ambiente. La probabilidad $P(x(n,m))$ será expresada con la regla de la probabilidad total, teniendo como resultado:

$$P(w_i|x(n,m)) = \frac{h_i[x(n,m)]}{\sum_{j=0}^{C-1} h_j[x(n,m)]} = \frac{1}{K} \cdot h_i[x(n,m)]$$

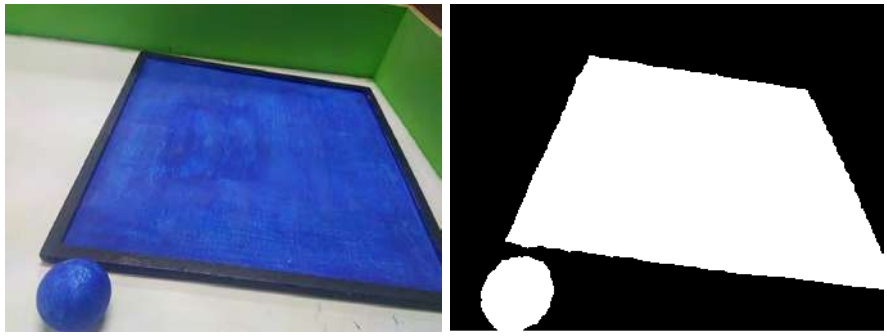
donde:

$$K = \sum_{j=0}^{C-1} h_j[x(n,m)]$$

En la Figura 3.5 se muestran los resultados de aplicar el Algoritmo 1 con entrenamiento a base de histograma, ésta segmentación fue aplicada a 2 imágenes similares a la Figura 3.3. Ambas segmentaciones para la Figura 3.5 (a) y (c) fueron realizadas con un valor $d = 2$, el resultado de ambas segmentaciones se muestra en la Figura 3.5 (b) y (d). Aunque no es posible verificar la calidad de la segmentación, se puede observar que es lo suficientemente eficaz para ambas imágenes a pesar de los cambios de iluminación, esta segmentación fue realizada en un tiempo de 0.0948 segundos con ambas imágenes de tamaño 640×480, lo cual la hace apropiada para ser realizada en tiempo real.



(a) **(b)**

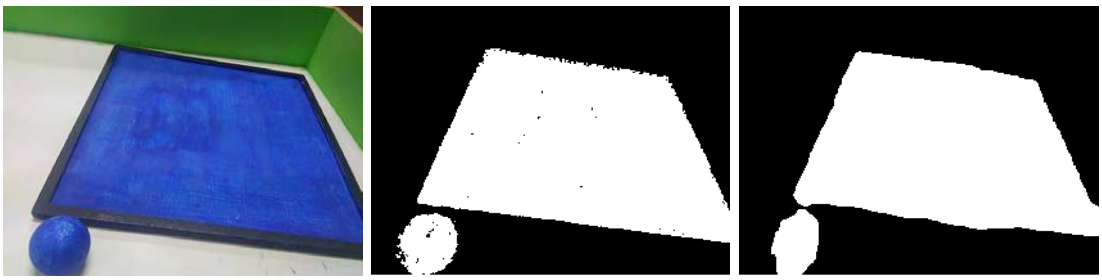


(c) **(d)**

Figura 3.5: Resultados de segmentación con coherencia espacial para la figura 3.3. Los pixeles blancos corresponden a la clase azul, mientras que los negros corresponden al resto de la imagen. En (a) se muestra la primera imagen original, en (b) los resultados de su segmentación. En (c) se muestra la segunda imagen original, en (d) los resultados de su segmentación.

Elegir un tamaño de ventana adecuado es crucial para conseguir una segmentación eficiente, en la Figura 3.6 se muestra este problema, en (b) se realizó la segmentación con un valor $d = 1$, donde se puede observar claramente que la coherencia espacial no fue suficiente, mientras que en (c) se realizó la segmentación con un valor $d = 12$, donde se puede observar que la segmentación no muestra la forma circular de la pelota azul.

Para mejorar la segmentación de la imagen, se propone reducir el espacio de color de los pixeles, este proceso consiste en reducir el número de valores que pueden representar colores para un pixel dado, por ejemplo, si el rango de valores que representan los canales R, G, B están en $[0, 255]$ para cada canal, el número de colores que se pueden representar es



(a) **(b)** **(c)**

Figura 3.6: Problema del tamaño d para la coherencia espacial. En (a) se muestra la imagen original, en (b) la segmentación con $d = 1$ y en (c) con $d = 12$.

16,777,216, si el espacio de color se reduce de manera que el rango sea de $[0, 63]$, el número de colores que se pueden representar es 262,144. Al reducir el espacio de color también se reduce el tamaño del histograma. Este proceso mejora la segmentación porque al reducir este espacio en una imagen, el número de vectores que se repiten en una posición dada del histograma aumenta, por ejemplo, si quisieramos almacenar los vectores $[0, 0, 255]$ y $[0, 0, 254]$, estos serían almacenados en posiciones diferentes para el histograma, en cambio, si reducimos este espacio en el rango $[0, 63]$, los vectores cambiarian a $[0, 0, 63]$ y $[0, 0, 63]$ por lo que serían almacenados en la misma posición del histograma.

3.5. Conclusiones

Se ha presentado el método SMDF extendido para dos o más clases. También se presentó la coherencia espacial y se demostraron las ventajas de aplicarla para mejorar la segmentación. También se mostró que el método SMDF extendido utilizando histogramas de color puede ser robusto ante cambios de iluminación y que puede ser llevada a cabo en centésimas de segundo, por lo que se considera apropiado para ser utilizado en tiempo real. En el capítulo 5 se muestran los resultados de aplicar esta segmentación con un robot de servicio.

Capítulo 4

Programación del robot de servicio

En este capítulo se describen los aspectos técnicos del robot de servicio, así como los algoritmos que fueron aplicados para realizar su tarea. La programación del robot consiste en implementar métodos que permiten al robot obtener información del ambiente como la forma de los objetos y su ubicación, con el objetivo de que el robot pueda desempeñarse de manera apropiada dentro del ambiente donde se llevarán a cabo los experimentos. Algunos de estos métodos fueron aplicados utilizando la librería OpenCV con el lenguaje de programación C++.

4.1. Especificaciones técnicas del robot de servicio

En la Figura 4.1 se muestra el robot con el que se llevarán a cabo los experimentos para esta tesis. Este robot fue construido para llevar a cabo tareas sencillas de movimiento como avanzar una distancia dada o girar un ángulo dado. El robot utiliza una cámara con la cual se pueden capturar imágenes del ambiente. El robot fue construido para realizar la tarea de tomar pelotas de colores y llevarlas a depósitos dentro de un ambiente controlado, con el objetivo de ser aplicado en una competencia de robots de servicio, la cual se describe en [com16]. Las especificaciones técnicas del robot son las siguientes:

- Cuenta con un dispositivo Odroid U3 que funciona como una computadora a bordo [odr18].

Figura 4.1: Robot de servicio.

- Cuenta con una cámara Logitech C920, la cual puede tomar imágenes a color de hasta 1920 × 1080 pixeles.
- Cuenta con una pinza con dos grados de libertad, uno para subir o bajar la pinza, y uno para abrir o cerrar la pinza.
- Utiliza un esquema de locomoción diferencial mediante 2 ruedas de tracción accionadas por dos motores de pasos independientes.
- Para darle estabilidad, el robot cuenta con una rueda loca en la parte delantera.
- Para el control de los motores de pasos de las ruedas, así como las pinzas, el robot cuenta con un microcontrolador arduino Mega 2560, el cual se conecta al odroid U3 mediante una conexión USB.

4.2. Ambiente real del robot de servicio

El ambiente real en el que estará operando el robot es mostrado en la Figura 4.2, el cual consiste en un tablero de madera con un tamaño de 100 cm ×150 cm ×10 cm. En este tablero el piso está pintado de color blanco, y las paredes de verde. En el tablero se

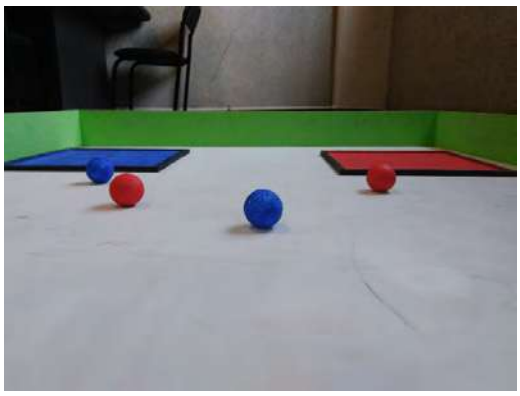


Figura 4.2: Ambiente del robot de servicio.

encuentran esferas de unicel de color rojo o azul con un diámetro de 4 cm. También se encuentran dos depósitos, uno es de color rojo y otro de color azul. Los depósitos tienen forma cuadrada de 30 cm $\times$30 cm, con bordes de 1 cm de ancho y 1 cm de alto de color negro. Este ambiente fue construido siguiendo los requisitos designados para la competencia de robots de servicio dado por [com16].

El robot debe realizar la tarea de tomar las pelotas de unicel y llevarlas una por una al depósito del mismo color que la pelota, llevando así las pelotas de color rojo al depósito rojo, y las pelotas de color azul al depósito azul. El número de pelotas que hay en el tablero puede variar dependiendo del tipo de experimento que se quiere realizar, algunos de los experimentos realizados en esta tesis fueron llevados a cabo con 2, 4 y 6 pelotas. El robot debe ser capaz de desempeñarse correctamente independientemente del número de pelotas que se encuentren en el tablero. La imagen capturada en la Figura 4.2 fue tomada por la cámara del robot de servicio, por lo tanto, este es el tipo de imágenes que estará procesando el robot. Se dice que el robot tendrá un desempeño correcto, si este es capaz de detectar de manera correcta las pelotas de colores, avanzar, tomar estas pelotas y llevarlas a sus depósitos de color correspondiente.

4.3. Instrucciones de movimiento para el robot de servicio

Gracias a la comunicación serial que se efectúa entre el Odroid y el arduino, es posible enviar instrucciones en forma de comandos hacia el robot de servicio. Estas instrucciones le permiten al robot realizar tareas de movimiento para desplazarse dentro de su entorno.

Comando	Respuesta	Descripción
$l\# < cmd >< nl >$	$t < nl >$	Si $< cmd >= 0$, apaga el led, si $< cmd >= 1$, enciende el led
$b < nl >$	$[01] < nl > t < nl >$	El robot regresa el estado del botón rojo del robot 0 si no esta presionado, 1 si está presionado
$a\# < d >< nl >$	$t < nl >$	Avanza $< d > mm$. Si $d < 0$, el robot retrocede.
$g\# < ang >< nl >$	$t < nl >$	Gira un ángulo de $< ang >$ grados. si $ang > 0$, gira a la izquierda (antihorario), si $ang < 0$, gira a la derecha (horario).
$s\# < valor >< nl >$	$t < nl >$	Sube la pinza un cierto valor. Su rango de operación es de -20 (abajo) a 100 (arriba). (0 es horizontal)
$c\# < valor >< nl >$	$t < nl >$	Cierra pinza. Su rango de operación es de 0 (cerrada) a 100 (abierta).
$m\# < valor >< nl >$	$t < nl >$	Control de bloqueo de los motores. Si $< cmd >= 0$, libera motores (ahorra energía). Si $< cmd >= 1$, bloquea motores (las ruedas no se mueven). Por omisión no se bloquean después de hacer un movimiento.
$v\# < v_i > \# < v_d >< nl >$	$t < nl >$	Fija velocidades de ruedas en mm/s. $< v_i >$ es la velocidad de la rueda izquierda. $< v_d >$ es la velocidad de la rueda derecha (visto el robot por atrás).

Tabla 4.1: Lista de comandos de comunicación para el robot de servicio.

En la Tabla 4.1 se muestran los comandos de comunicación para el robot de servicio que se utiliza para esta tesis. En esta tabla, se muestra el comando que se envía del Odroid

al Arduino, la respuesta que devuelve el Arduino y la descripción de la acción que ejecuta el robot de acuerdo con el comando. En la tabla se utilizan las siguientes convenciones [Romero16]:

- # Representa un espacio.
- $< nl >$ Representa una nueva línea.
- $< d >$,$< ang >$,$< valor >$,$< v_i >$ y $< v_d >$ representan números reales positivos o negativos.

4.4. Calibración de la cámara

Para calcular el ángulo y la distancia de las pelotas y depósitos capturados por la cámara, el primer paso es calibrar la cámara del robot. Adrian Kaheler y Gary Bradski definen la calibración como un proceso que nos permite encontrar un modelo de la geometría de la cámara y un modelo de distorsión de los lentes de la cámara [Kaehler16]. La motivación de llevar a cabo la calibración está relacionado con los cambios internos de fabricación que ocurren al construir las cámaras. Estas variaciones de manufactura durante esta fabricación provoca que los parámetros intrínsecos de la cámara sean diferentes para cada cámara, así como la distorsión radial de cada lente. A continuación se describe el modelo de la cámara así como la distorsión radial.

4.4.1. Modelo de la cámara

El modelo más simple de una cámara es el modelo *pinhole*, el cual se describe con detalle en [Prince12]. Este modelo consiste en una caja que contiene un pequeño agujero en la parte frontal. Los rayos de luz provenientes de un objeto del mundo real pasan a través de este agujero para formar una imagen invertida en la parte trasera de la caja, la cual es considerada como el plano de la imagen. En la Figura 4.3 se muestra la forma en la que se dibuja una imagen usando este modelo.

El modelo completo de una cámara pinhole está expresado por dos conjuntos de parámetros: 1) los parámetros intrínsecos, los cuales describen la cámara en sí; 2) los

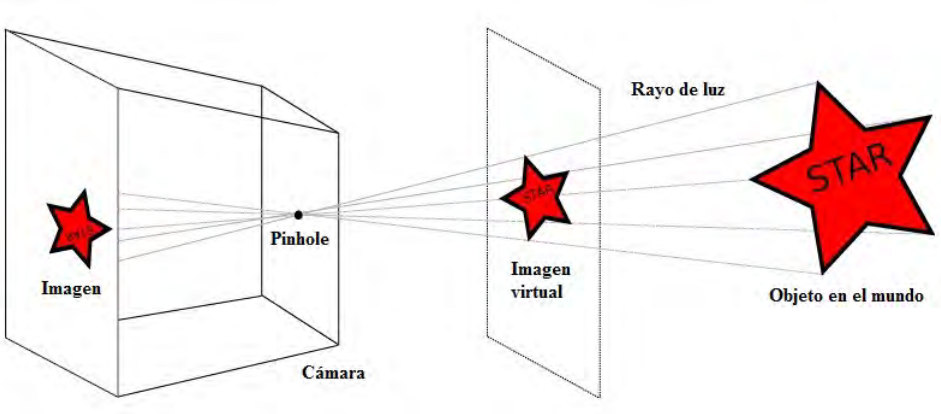


Figura 4.3: Modelo de la cámara pinhole.

parámetros extrínsecos, los cuales describen la posición y rotación de la cámara con respecto de un sistema de referencia global. Los parámetros intrínsecos son expresados en la matriz (4.1):

$$\Lambda = \begin{bmatrix} \phi_{\hat{x}} & \gamma & \delta_{\hat{x}} \\ 0 & \phi_{\hat{y}} & \delta_{\hat{y}} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

donde:

- $\phi_{\hat{x}}$ y $\phi_{\hat{y}}$ son la distancia focal de la cámara en el eje horizontal $\hat{x}$ y el eje vertical $\hat{y}$.
- $\delta_{\hat{x}}$ y $\delta_{\hat{y}}$ definen el centro óptico de la cámara, los cuales son expresados en unidades de pixeles. Para una cámara perfectamente centrada, estos parámetros deben estar en el centro de la imagen de la cámara, por ejemplo, para una imagen de 640×480, los valores del centro óptico serían $\delta_{\hat{x}} = 320$ y $\delta_{\hat{y}} = 240$. Sin embargo, en la práctica es bastante complicado lograr la fabricación de una cámara con un centro óptico perfectamente centrado, por lo tanto, estos valores son variables.
- γ es un parámetro de desviación (en inglés conocido como *skew*), el cual representa un torcimiento del plano de la imagen de la cámara.

En la Figura 4.4 se muestra la distancia focal y el centro óptico de la cámara. El centro óptico es considerado como el punto que se encuentra en el origen del sistema de

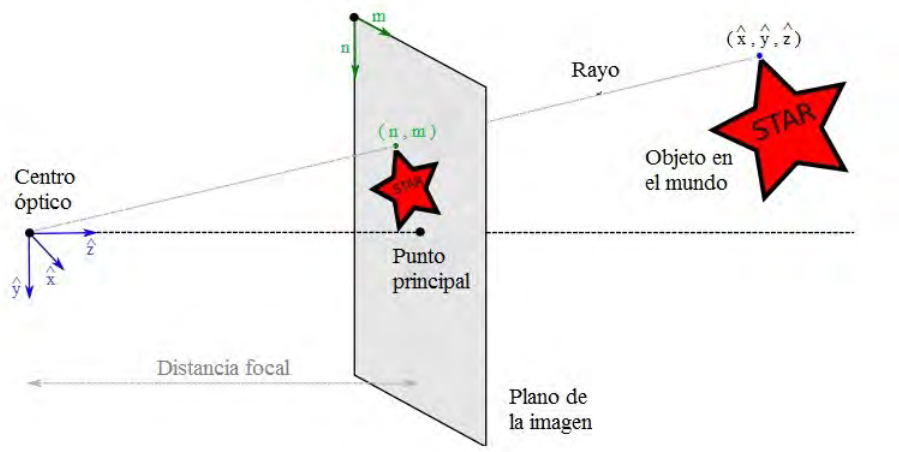


Figura 4.4: Parámetros intrínsecos de la cámara. El centro óptico es considerado como el punto que se encuentra en el origen del sistema de coordenadas del ambiente real. La distancia focal es considerada la distancia que hay entre el centro óptico y el plano de la imagen.

coordenadas del ambiente real. La distancia focal es considerada la distancia que hay entre el centro óptico y el plano de la imagen.

Los parámetros extrínsecos de la cámara son expresados por una matriz de rotación:

$$\Omega = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \tag{4.2}$$

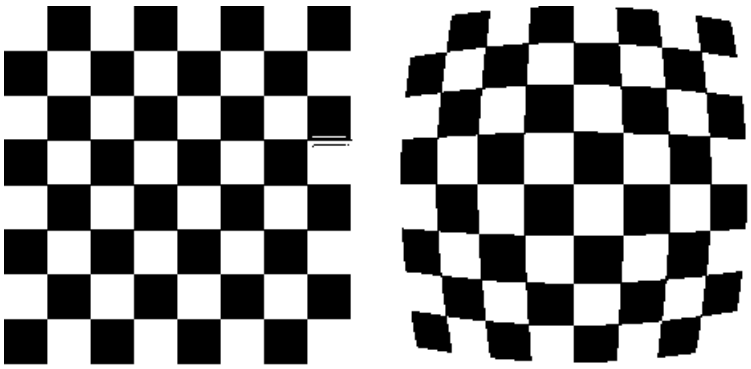
y un vector de traslación:

$$\tau = \begin{bmatrix} \tau_{\hat{x}} \\ \tau_{\hat{y}} \\ \tau_{\hat{z}} \end{bmatrix} \tag{4.3}$$

Por lo tanto, para encontrar nuestro modelo de la cámara, es necesario encontrar la matriz intrínseca Λ, la matriz de rotación Ω y el vector de traslación τ

4.4.2. Coeficientes de distorsión

El modelo de la cámara pinhole es un modelo simple y útil para entender como funciona una cámara desde un punto de vista básico, sin embargo, como se describe por Simon Prince [Prince12], las cámaras del mundo real raramente están basadas en el modelo pinhole. Esto es debido a que las cámaras utilizan lentes, los cuales suelen tener una forma curva, lo cual provoca que los rayos de luz de un objeto al pasar por el lente son desviados terminando en una posición diferente en el plano de la imagen.



(a) **(b)**

Figura 4.5: Distorsión radial de una imagen. En (a) se muestra la imagen del objeto original sin distorsión, en (b) se muestra la imagen del objeto sometida a distorsión radial.

En la Figura 4.5 se muestra este problema, donde se puede observar la imagen de un tablero cuadrado. En (a) se muestra la imagen del tablero sin recibir ningún tipo de distorsión, en (b) se muestra la imagen del mismo tablero sometida a distorsión radial. Si queremos extraer información como la posición de las esquinas de este tablero en coordenadas del mundo real, al hacerlo con (a), obtendremos las coordenadas correctas, en cambio, si lo hacemos con (b) tendremos coordenadas equivocadas que pueden provocar errores de cálculo al momento de obtener un ángulo o distancia.

La distorsión radial puede ser modelada como una función polinómica de una distancia r desde el centro óptico de la imagen al pixel en la posición $(\hat{x}, \hat{y})$. En una cámara normalizada, para la cual la distancia focal es uno y se asume que el origen del sistema de coordenadas $2D$ se encuentra en el centro de la imagen, los puntos finales de la imagen $(\hat{x}', \hat{y}')$

son expresados como funciones de los puntos originales $(\hat{x}, \hat{y})$ por las ecuaciones [Prince12]:

$$\begin{aligned} \hat{x}' &= \hat{x}(1 + \beta_1 r^2 + \beta_2 r^4) \\ \hat{y}' &= \hat{y}(1 + \beta_1 r^2 + \beta_2 r^4) \end{aligned} \tag{4.4}$$

donde los parámetros β_1 y β_2 controlan el grado de distorsión. Este modelo es útil para describir un conjunto de distorsiones que usualmente se presentan en la mayoría de los lentes de amplia apertura.

4.4.3. Cálculo del modelo de la cámara y coeficientes de distorsión

Para calcular los parámetros previamente descritos, para esta tesis se utilizó la librería OpenCV en C++, la cual ya cuenta con las herramientas necesaria para la estimación de estos parámetros [ope18].

Para llevar a cabo la calibración de una cámara con OpenCV, se llama a la función cv::calibrateCamera(), la cual devuelve los parámetros dados por las ecuaciones (4.1),(4.2),(4.3) y (4.4).

4.5. Vista de pájaro

La vista de pájaro es un tipo de transformación de perspectiva comunmente usada en la navegación de los robots. La vista de pájaro consiste en convertir la vista de la cámara de un robot en una especie de "vista desde arriba", simulando la vista que tiene un pájaro mientras está volando [Kaehler16].

La ventaja de este tipo de vista es que al aplicar la transformación, para cada pixel de la imagen, se obtienen posiciones correspondientes al ambiente del robot en unidades métricas. Por lo tanto, para nuestro problema será posible conocer la posición de las pelotas de unicel, así como los depósitos. Esta vista solo puede ser aplicada en un ambiente real donde el piso es plano, ya que un piso irregular provocará cálculos de distancias erróneas. Dado que el piso es plano, se asume que el valor $\hat{y}$ del sistema de coordenadas del ambiente real no varia y se puede fijar a un valor constante, en este caso 0.

Para simplificar la lectura de esta sección, el ambiente real será representado en un plano bidimensional en el eje $\hat{x}$ y $\hat{z}$ los cuales representan anchura y profundidad, en la

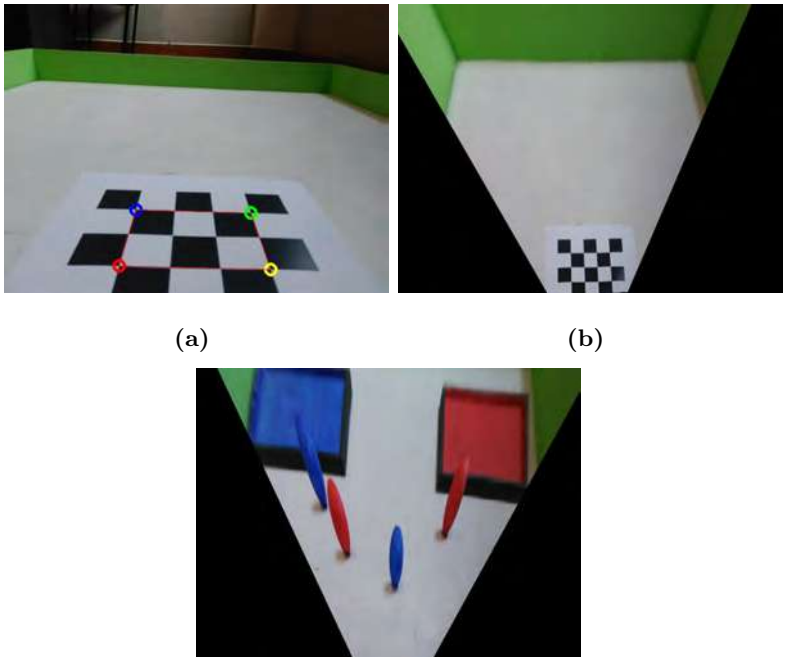
Figura 4.6 se muestran estas coordenadas. El eje $\hat{y}$ que representa la altura será omitido ya que no contiene información relevante que pueda ser utilizada para nuestro propósito.

Figura 4.6: Coordenadas $(\hat{x}, \hat{z})$ del ambiente del robot.

Para conseguir la vista de pájaro, es necesario tener la matriz intrínseca y los parámetros de distorsión de la cámara, los cuales fueron explicados en la sección anterior. El proceso consiste en calcular una matriz de homografía de 3×3. En esta tesis se explica como aplicar la vista de pájaro utilizando las librerías de OpenCV en C++. La matriz de homografía puede ser computada con los siguientes pasos [Kaehler16]:

- Leer los parámetros intrínsecos de la cámara y los parámetros de distorsión.
- Encontrar un objeto conocido en la imagen de la cámara (en este caso un tablero de ajedrez), el objeto debe estar colocado en el piso y se deben detectar al menos 4 puntos específicos del tablero con precisión subpixel. La ubicación de los 4 puntos en el tablero se conocen con precisión.

- Calcular una matriz de homografía H utilizando la función cv::getPerspectiveTransform(), a partir de las coordenadas de los 4 puntos en el tablero y en la imagen.
- Utilizar la función cv::warpPerspective() para obtener la vista de pájaro con respecto al piso.



(a) **(b)**

(c)

Figura 4.7: Transformación de vista de pájaro. En (a) se muestra la imagen utilizada para obtener la Homografía, en (b) se muestra la imagen resultante de aplicar la transformación de vista de pájaro para (a) y en (c) se muestra la imagen resultante al aplicar la transformación de vista de pájaro para la Figura 4.2.

En la Figura 4.7 se muestran los resultados de aplicar la vista de pájaro, en (a) se muestra la imagen con el tablero de ajedrez y sus 4 puntos utilizados para calcular la matriz de homografía, en (b) se muestran los resultados de aplicar esta transformación para (a) y en (c) se muestran los resultados de aplicar esta transformación para la Figura 4.2. En esta vista, calcular la distancia a la que se encuentra un objeto en el mundo real ahora es posible, este tema será explicado a continuación.

La vista de pájaro es un proceso de calibración que obtiene una correspondencia de

los puntos de la imagen con las coordenadas del ambiente del robot. Esta calibración debe ser realizada con la mayor precisión posible ya que una calibración errónea puede provocar que al procesar los puntos de cada pixel se obtengan coordenadas que no corresponden de forma correcta al ambiente, provocando cálculos de distancia con errores de varios centímetros.

Para verificar si esta calibración es correcta se puede calcular la distancia que hay entre 2 puntos de la imagen utilizando la matriz de homografía y se compara con la distancia real en el ambiente, si la distancia de estos 2 puntos tiene un error menor a un valor mínimo, esta puede ser considerada como una calibración correcta.

En la Figura 4.8 se muestra el método para verificar la precisión de esta calibración, en esta figura se muestra un tablero de ajedrez para la cual se seleccionaron 2 de sus esquinas, la distancia entre estas dos esquinas es de $14.42cm$. Posteriormente se realizó el cálculo de la distancia utilizando la imagen de la vista de pájaro, la distancia calculada para este caso es de $14.34cm$, lo que nos da un error de $0.08cm$, siendo esta una calibración con un error menor a 1 milímetro.

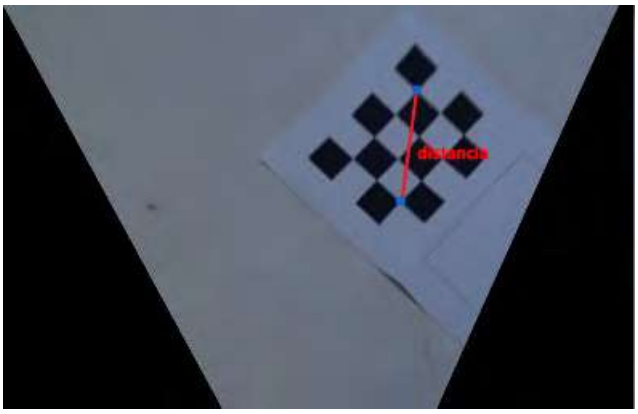


Figura 4.8: Cálculo de error para la vista de pájaro. Se realiza el cálculo de la distancia entre las 2 esquinas del tablero seleccionadas en la figura, después se verifica esta distancia con la distancia real de las esquinas. El margen de error para este cálculo es de $0.08cm$.

4.5.1. Cálculo de distancia y ángulo para un objeto en el ambiente real

Una vez que se obtiene la posición de los objetos en coordenadas del mundo real, se puede calcular el ángulo y la distancia de un objeto con respecto al robot de servicio, de

esta manera el robot puede dirigirse a un objeto como una pelota o depósito, se le puede dar la instrucción de girar el ángulo dado de tal manera que le permita al robot alinearse con el objeto, posteriormente, se le da la instrucción de avanzar en línea recta la distancia dada de manera que el objeto se encuentre en medio de las pinzas del robot, para así poder tomar el objeto con las pinzas. Para lograr este cometido, es necesario calcular el ángulo y la distancia del objeto con respecto al sistema de referencia del robot.

Para llevar a cabo el cálculo del ángulo y la distancia, posicionaremos el centro de coordenadas del mundo real en el centro de rotación del robot de servicio, el cual se encuentra en medio de sus ruedas, luego haremos el cálculo de la matriz de homografía donde los puntos del tablero corresponden con este centro de coordenadas.

En la Figura 4.9 (a) se muestra el tablero de ajedrez utilizado para calcular la homografía, en la Figura 4.9 (b) se muestra el área donde se coloca el robot, de manera que este se encuentre lo más alineado posible. Las unidades mostradas tanto en (a) como en (b) corresponden a las coordenadas del ambiente real en centímetros, estos puntos son explicados a continuación:

- La posición (0,0) corresponde al origen el cual esta situado en el centro del robot de servicio.
- Las posiciones (9.5,0) y (-9.5,0) corresponden a la posición de las ruedas derecha e izquierda del robot.
- La posición (0,27) corresponde al centro de las pinzas del robot.
- Las posiciones (6,34), (-6,34), (6,42) y (-6,42) corresponden a las esquinas del tablero seleccionadas para hacer el cálculo de la Homografía.

Al calcular la matriz de homografía utilizando estos puntos, cada vez que se procese un punto de la imagen capturada por la cámara del robot, tendremos como resultado un punto (x, z) con respecto al centro del robot de servicio, ahora solo se calcula la distancia euclidiana y el ángulo entre el punto dado y el origen, de la forma:

$$\begin{aligned} d &= \sqrt{\hat{x}^2 + \hat{z}^2} \\ \theta &= \arcsin(\tfrac{\hat{x}}{d}) \end{aligned} \tag{4.5}$$

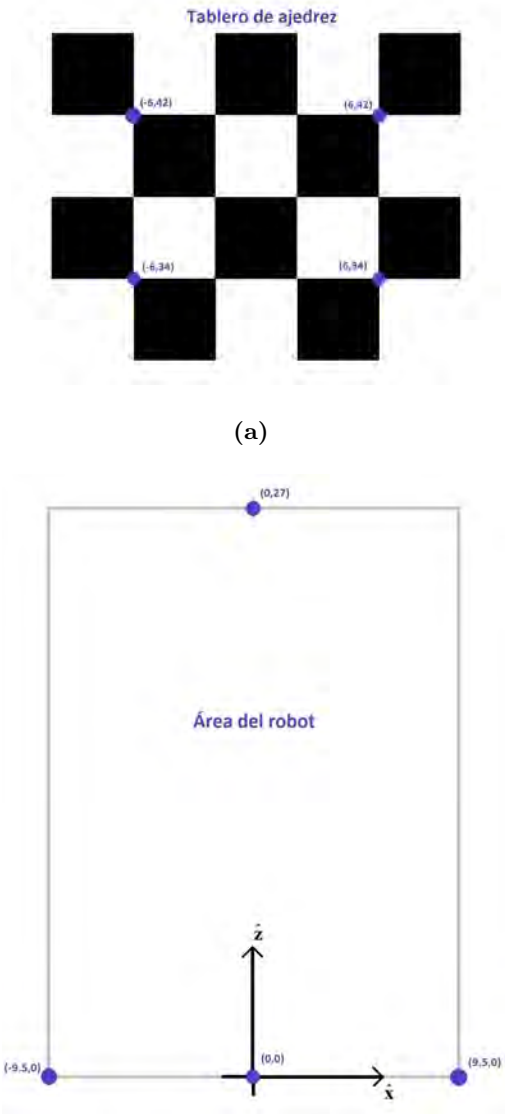


Figura 4.9: Tablero y área utilizados para calcular la Homografía. En (a) se muestra el tablero de ajedrez utilizado para calcular la matriz de homografía, en (b) se muestra el área donde se posiciona el robot.

Para que el cálculo de la distancia hacia un punto $(\hat{x}, \hat{y})$ del plano de la imagen sea correcto, este punto debe corresponder al piso del tablero, es decir, para calcular la distancia de una pelota hacia el robot, se debe procesar el punto que se encuentre en la base de la pelota justo donde toca el piso. En la Figura 4.10 se muestra un ejemplo del cálculo

de la distancia y ángulo aplicado a la Figura 4.7 (c), estos valores fueron calculados con (4.5) para los puntos color verde mostrados en la Figura 4.10.

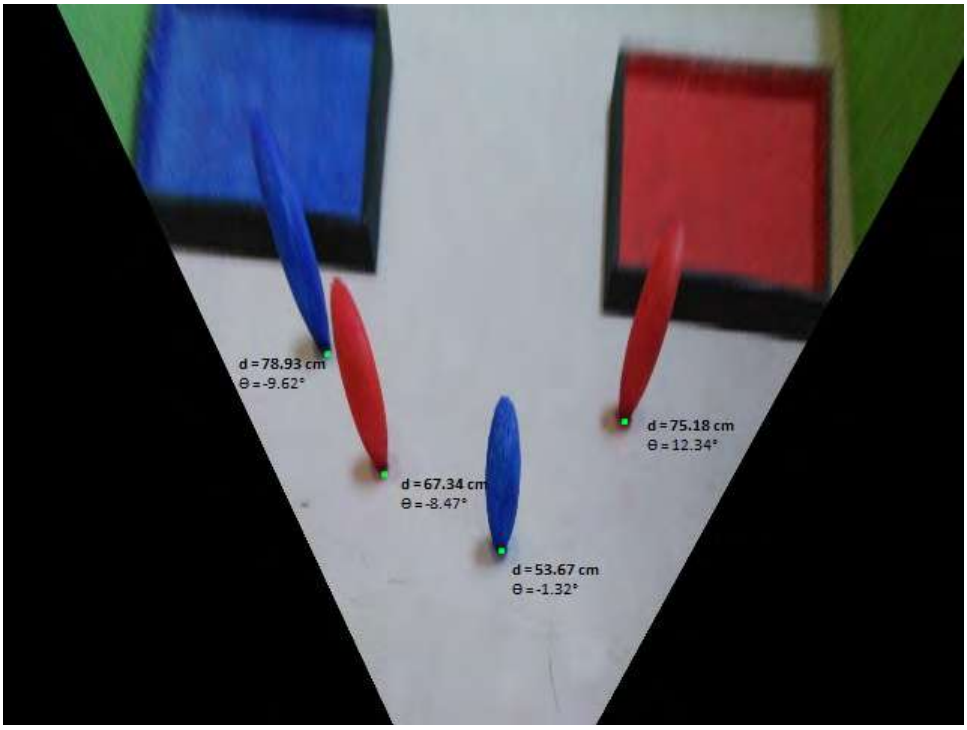


Figura 4.10: Aplicación de la vista de pájaro al ambiente real del robot de servicio y cálculo del ángulo y distancia.

4.6. Detección de cúmulos

Durante el desarrollo de esta tesis se presentó el problema de encontrar cierta cantidad de pelotas o depósitos en una imagen dada. En la Figura 4.11 se muestra el ejemplo de segmentación para el color azul en una imagen dada. En la Figura 4.11 (b) se muestra la imagen resultante de la segmentación, en la cual se puede observar un conjunto de cúmulos, para esta tesis definimos un cúmulo como un conjunto de pixeles interconectados entre sí, de manera que cada pixel que pertenece al cúmulo tiene al menos un pixel vecino que también pertenece al cúmulo alrededor de el. Cada cúmulo corresponde a un objeto, en esta figura se muestran 3 cúmulos de los cuales 2 pertenecen a pelotas de unicel y 1 al depósito.

Aunque estos cúmulos mostrados en la Figura 4.11 (b) pueden ser detectados a

simple vista con el ojo humano, llevar a cabo esta tarea es más complicado para una computadora ya que el procedimiento se debe realizar pixel por pixel. Por otro lado, es necesario considerar dos problemáticas que se presentan al momento de proponer una solución:

(a) **(b)**

Figura 4.11: Ejemplo de segmentación del color azul para una imagen dada,.

- El número de cúmulos que pueden aparecer en una imagen segmentada puede variar. Dependiendo del experimento que se realiza y las acciones que toma el robot, este puede encontrarse enfrente de un número n de pelotas y depósitos, donde n es un número desconocido.
- Dependiendo de la calidad y eficacia del segmentador, puede ocurrir la situación en la que aparezcan datos erróneos en la segmentación, estos datos se presentan en forma de pequeños cúmulos con un tamaño de apenas unos pocos pixeles (a veces de un solo pixel). Estos datos erróneos no deben ser procesados pueden mostrar objetos inexistentes en el ambiente.

Teniendo en cuenta estas dificultades, se presenta el método que fue propuesto para llevar a cabo la detección de cúmulos correcta.

4.6.1. Algoritmo para encontrar cúmulos

En esta tesis se propone un algoritmo llamado `Encontrar_Cúmulos`. Este algoritmo tiene como objetivo encontrar un conjunto de cúmulos que se encuentran dentro de una imagen dada, considerando a su vez, las 2 problemáticas presentadas al inicio de esta sección.

Primero consideremos que se tiene una imagen binaria I la cual es el resultado de una segmentación de color, los pixeles pertenecientes a la clase del color segmentado están dados por el valor 255 (color blanco), mientras que los pixeles que no pertenecen están dados por el valor 0 (color negro).

El algoritmo `Encontrar_Cúmulos` es mostrado en el **Algoritmo 2**, en la entrada del algoritmo se tiene una imagen I, un conjunto vacío D y un conjunto vacío M. El algoritmo consiste en llevar a cabo un recorrido de toda la imagen hasta encontrar un pixel blanco (cuyo valor sea igual a 255), una vez que este pixel fue encontrado, se asume que se encontró un cúmulo, por lo que se crea una cola llamada D_k y ésta es agregada a D, D_k es una cola que guarda los puntos (n, m) que representan la posición de los pixeles blancos encontrados en cada cúmulo. D_k tiene la función $ingresar()$ que agrega un punto (n, m) al final de la cola, también tiene la función $remover()$ que elimina el primer punto que fue insertado en la cola.

Algoritmo 2: Algoritmo `Encontrar_Cúmulos`

Datos: Una imagen I, un conjunto $D = \{\}$, un conjunto $M = \{\}$
Resultado: M
Hacer $k = 0$
para $n = 0$ ***hasta*** $n_r - 1$ **hacer**
  para $m = 0$ ***hasta*** $n_c - 1$ **hacer**
    si $I(n, m) = 255$ **entonces**
      Crear la cola D_k
      $ingresar(D_k, (n, m))$
      Agregar D_k a D
      $M_k = Procesar\_Cola(I, D_k, k)$
      Agregar M_k a M
      $k = k + 1$
    fin
  fin
fin
devolver M

Algoritmo 3: Algoritmo `Procesar_Cola`

Datos: Una imagen I, una cola $D_k = \{(r, c)\}$, un conjunto $M_k = \{\}$, k

Resultado: M_k

Hacer $M_k = \{(r, c)\}$

mientras $noEstaVacia(D_k)$ **hacer**
- $(r, c) = remover(D_k)$
- **si** $I(r-1, c) = 255$ **entonces**
  - **si** $(r-1, c)$ *NO está en* M_k **entonces**
    - $ingresar(D_k, (r-1, c))$
    - Agregar $(r-1, c)$ a M_k
  - **fin**
- **fin**
- **si** $I(r, c-1) = 255$ **entonces**
  - **si** $(r, c-1)$ *NO está en* M_k **entonces**
    - $ingresar(D_k, (r, c-1))$
    - Agregar $(r, c-1)$ a M_k
  - **fin**
- **fin**
- **si** $I(r+1, c) = 255$ **entonces**
  - **si** $(r+1, c)$ *NO está en* M_k **entonces**
    - $ingresar(D_k, (r+1, c))$
    - Agregar $(r+1, c)$ a M_k
  - **fin**
- **fin**
- **si** $I(r, c+1) = 255$ **entonces**
  - **si** $(r, c+1)$ *NO está en* M_k **entonces**
    - $ingresar(D_k, (r, c+1))$
    - Agregar $(r, c+1)$ a M_k
  - **fin**
- **fin**
- Hacer $I(r, c) = 0$

fin

devolver M_k

Una vez que se encuentra un pixel blanco inicial en la imagen, se lleva a cabo una búsqueda de todos los pixeles blancos que se encuentran cerca de este pixel inicial, el método *Procesar_Cola*() mostrado en el **Algoritmo 3** es el encargado de realizar esta tarea. Este método procesa D_k, aplicando la función *remover*() que permite obtener el

punto inicial (r, c) de la cola y posteriormente lo elimina, después se verifican los pixeles que se encuentran en las posiciones $(r-1, c)$,$(r, c-1)$,$(r+1, c)$,$(r, c+1)$ respectivamente, todos los pixeles blancos son agregados a D_k utilizando la función $ingresar()$, al mismo tiempo son agregados al conjunto M_k (verificando que no se repitan). Después, el pixel de la imagen en $I(r, c)$ se hace negro (se cambia su valor por 0), el motivo de que este pixel se haga negro es para que no sea procesado más de una vez. Por último, el proceso se repite hasta que la cola D_k está vacía.

El método `Procesar_Cola` devuelve el conjunto M_k el cual contiene todos los puntos que pertenecen a un cúmulo. El algoritmo `Encontrar_Cúmulos` devuelve el conjunto M el cual contiene todos los subconjuntos M_k. Si el número de elementos que contiene el subconjunto M_k es demasiado pequeño, este puede ser descartado, siendo considerado como un dato erróneo.

4.6.2. Calcular centro y tamaño de cúmulos

Tener el conjunto de pixeles pertenecientes a cada cúmulo es solo el primer paso de nuestro detector de cúmulos, ya que nuestro verdadero objetivo para el robot es conocer en que posición se encuentra el cúmulo, y que tamaño tiene. Por lo tanto, el siguiente paso es calcular la posición del centro del cúmulo, así como su tamaño.

Para lograr esto, calculamos el centro de masa para cada conjunto de puntos pertenecientes a los cúmulos encontrados. Por lo tanto, definimos cada subconjunto $M_k \in M$ como $M_k = \{(r_0, c_0), (r_1, c_1), ..., (r_{N_k-1}, c_{N_k-1})\}$, posteriormente, aplicamos (4.6):

$$\begin{aligned} \bar{r}_k &= \textstyle\sum_{i=0}^{N_k-1} \frac{r_i}{N_k-1} \\ \bar{c}_k &= \textstyle\sum_{i=0}^{N_k-1} \frac{c_i}{N_k-1} \end{aligned} \tag{4.6}$$

donde N_k es el número de elementos que contiene el conjunto M_k. Esto nos entrega como resultado un punto $(\bar{r}_k, \bar{c}_k)$ el cual corresponde al centro de masa para el cúmulo k.

Para determinar el tamaño de cada cúmulo, se puede hacer el cálculo de la matriz

de covarianza para cada conjunto de datos M_k, el cual está dado por (4.7):

$$\begin{aligned} \sigma_{k_{rr}} &= \sum_{i=0}^{N_k-1} \frac{(r_i-\bar{r}_k)^2}{N_k} \\ \sigma_{k_{rc}} = \sigma_{k_{cr}} &= \sum_{i=0}^{N_k-1} \frac{(r_i-\bar{r}_k)(c_i-\bar{c}_k)}{N_k} \\ \sigma_{k_{cc}} &= \sum_{i=0}^{N_k-1} \frac{(c_i-\bar{c}_k)^2}{N_k} \end{aligned} \tag{4.7}$$

por lo tanto, la matriz de covarianza es expresada de la siguiente forma:

$$\Sigma_k = \begin{bmatrix} \sigma^2_{k_{rr}} & \sigma^2_{k_{rc}} \\ \sigma^2_{k_{cr}} & \sigma^2_{k_{cc}} \end{bmatrix} \tag{4.8}$$

La matriz de covarianza representa la forma en la que están dispersos un conjunto de datos que siguen una distribución de probabilidad [Prince12]. Si asumimos que estos datos siguen una distribución normal, entonces la matriz de covarianza puede tomar 3 formas diferentes las cuales son:

- Matriz de covarianza esférica: representada con un círculo, como se muestra en las Figuras 4.12 (a) y (b).
- Matriz de covarianza diagonal Σ_{diag}: representada con una elipse sin rotación, como se muestra en las Figuras 4.12 (c) y (d).
- Matriz de covarianza completa Σ_{full}: representada con una elipse con un valor de rotación dado, como se muestra en las Figuras 4.12 (e) y (f).

Por lo tanto, la información que debemos extraer de estas matrices es el ancho y alto de la elipse, en caso de que sea una matriz esférica el ancho y alto son iguales, y su rotación dada la cual es 0 si es esférica o diagonal y diferente de 0 en el caso opuesto.

Para extraer esta información se puede llevar a cabo la descomposición de la covarianza, donde se concluye que una matriz de covarianza completa puede expresarse como un producto de una matriz de rotación R y una matriz de covarianza diagonal Σ'_{diag}, de la siguiente forma [Prince12]:

$$\Sigma_{full} = R^T \Sigma'_{diag} R$$

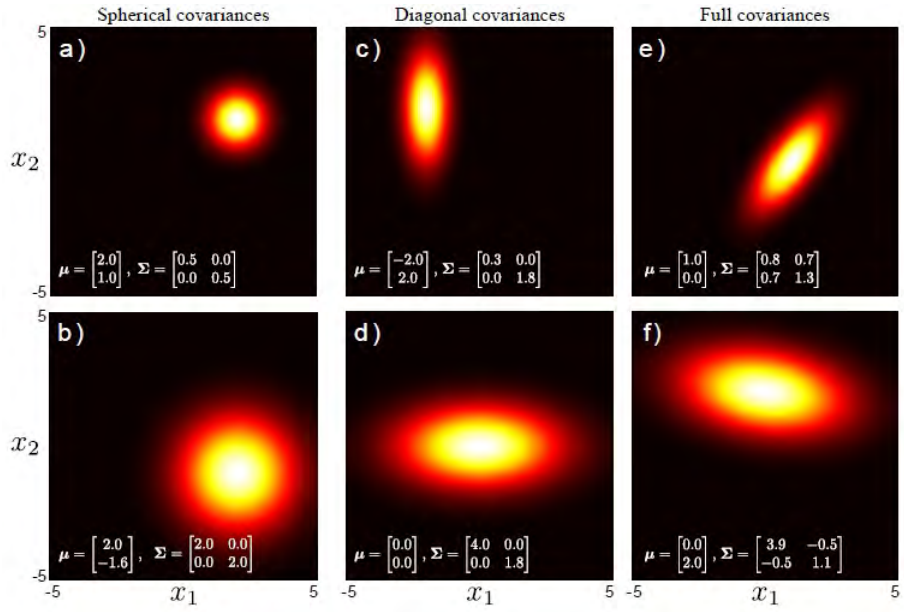


Figura 4.12: Formas de matrices de covarianza (fuente: [Prince12]).

Teniendo entendido esto, es posible encontrar R y Σ'_{diag} utilizando el método de descomposición de valores singulares. La matriz R es una matriz de rotación de la forma:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

donde θ es el ángulo de rotación de la elipse. La matriz Σ'_{diag} es una matriz diagonal de la forma:

$$\Sigma'_{diag} = \begin{bmatrix} \sigma'^2_{rr} & 0 \\ 0 & \sigma'^2_{cc} \end{bmatrix}$$

Por último, el ancho y alto de la elipse puede ser obtenido calculando la desviación estandar de cada valor y multiplicandola por 3, el ancho de la elipse está dado por $3\sigma'_{rr}$ y el alto está dado por $3\sigma'_{cc}$. El valor 3 se escogió siguiendo la regla tres sigma introducida por Friedrich Pukelsheim en [Pukelsheim94], la cual expresa que con un valor de 3σ se abarcan hasta un 99.7 % de los datos que se encuentran dentro de una distribución normal.

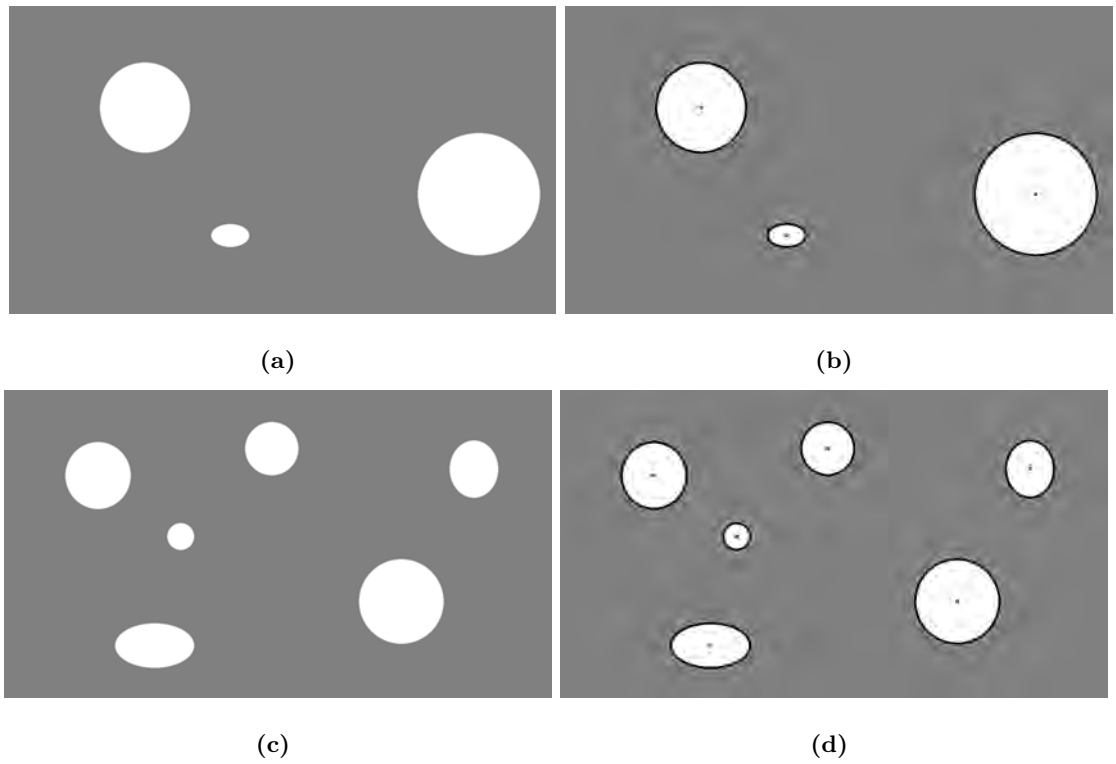


Figura 4.13: Resultados del algoritmo para encontrar cúmulos. En (a) y (c) se muestran las imágenes originales, en (b) y (d) se muestran los cúmulos detectados.

4.6.3. Resultados del algoritmo `Encontrar_Cúmulos`

En las siguientes figuras se muestran los resultados de aplicar el algoritmo para encontrar cúmulos, en (a) y (c) se muestran 2 imágenes binarias originales, en (b) y (d) se muestra el resultado de aplicar el algoritmo `Encontrar_Cúmulos`. Estas figuras fueron creadas utilizando software de edición de imágenes, esto con el fin de poder realizar pruebas controladas, para posteriormente aplicarlas en el robot de servicio.

En la Figura 4.13 se muestran estos resultados, donde (a) y (c) son imágenes binarias creadas con los valores $\{128, 255\}$, en (b) y (d) se muestra la detección de los cúmulos para cada imagen, donde se dibuja un pequeño círculo en el centro de cada cúmulo, y una elipse para mostrar su tamaño, la cual es dibujada utilizando las herramientas de openCV. Se puede observar que el algoritmo funciona para imágenes con 3 y 6 cúmulos.

4.6.4. Detección de forma del cúmulo para diferenciar depósitos y pelotas

Una vez que se conoce el número de cúmulos detectados, así como la posición y tamaño de estos mismos, es necesario conocer cuales cúmulos pertenecen a una pelota y cuales a un depósito. En la Figura 4.10 se puede observar que la vista de pájaro deforma los pixeles de las pelotas de manera que éstas parecen elipses alargadas, mientras que el depósito tiene una figura más cuadrada en comparación con las pelota, si calculamos la proporción del largo y ancho de un cúmulo, podemos observar que para el caso de una pelota, la proporción es mucho mayor que para el caso de un depósito.

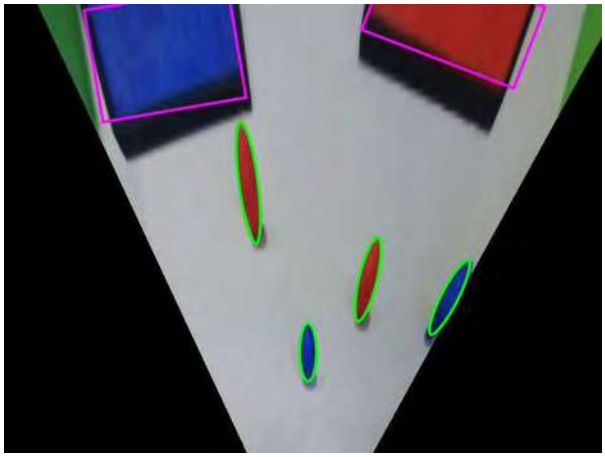


Figura 4.14: Resultados del algoritmo para encontrar cúmulos con detección de forma.

De esta manera, para determinar si un cúmulo corresponde a una pelota o un depósito, calculamos su proporción del largo y ancho donde dividimos el largo entre el ancho y evaluamos si esa proporción es mayor a un valor τ, si este es mayor, se considera que es una pelota, en caso contrario, se considera que es un depósito.

En la Figura 4.14 se muestra el resultado de aplicar el algoritmo para encontrar cúmulos con el ambiente real del robot, también se muestra el resultado de aplicar el método para distinguir las pelotas de los depósitos, donde se utilizo un valor $\tau = 1.5$, las pelotas fueron dibujadas con elipses verdes, mientras que los depósitos fueron dibujados con rectángulos de color púrpura.

4.7. Segmentación de color para el ambiente del robot de servicio

La segmentación de color para el ambiente del robot de servicio se llevó a cabo para un total de 5 clases, la clase 0 son las pelotas rojas y el depósito rojo, la clase 1 son las pelotas azules y depósito azul, la clase 2 es el piso blanco, la clase 3 es la pared verde del tablero y por último la clase 4 es todo lo que no pertenece a las clases anteriores, la cual será nombrada fondo.

Para llevar a cabo el entrenamiento del segmentador, se creó un método que permite al robot hacer el entrenamiento en tiempo real al inicio del experimento, esto con el objetivo de contar con un entrenamiento adecuado a los cambios de iluminación del ambiente que estén presentes durante el experimento. Este método será explicado a continuación.

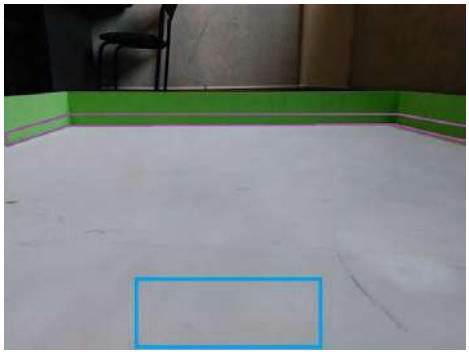
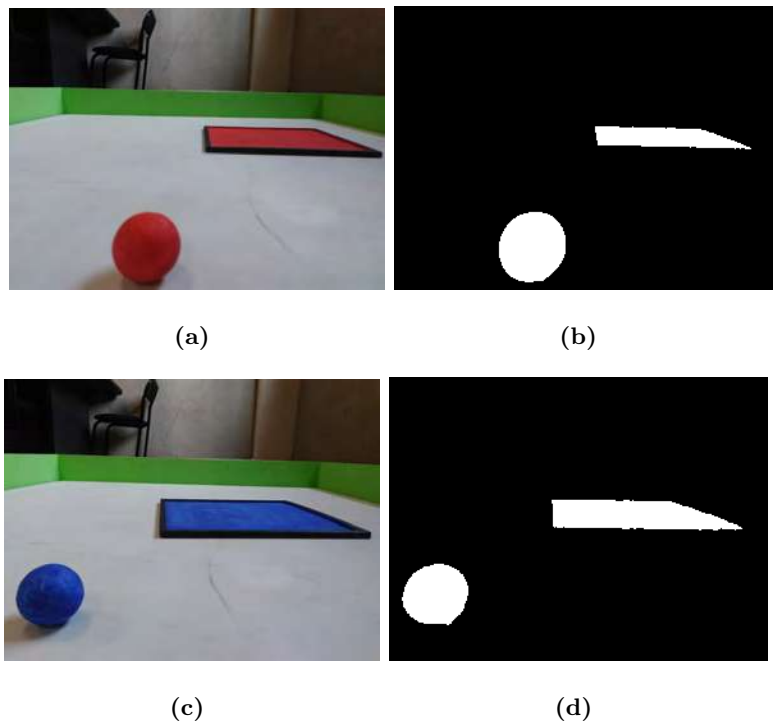


Figura 4.15: Entrenamiento para la clase 2 y 3.

Para realizar un entrenamiento para las clases 2, 3 y 4 se toma una imagen de muestra del ambiente del robot donde se observe el piso blanco y la pared verde del tablero, como en la Figura 4.15, La zona central inferior mostrada en el rectángulo azul de la figura puede ser tomada como una muestra de color blanco con la cual se construye un histograma para la clase 2. Posteriormente, se lleva a cabo una segmentación de la clase 2 empezando desde la parte inferior central de la imagen recorriendo todos los pixeles hasta los bordes del piso del tablero, de manera que podemos asumir que los pixeles cercanos que están arriba de estos bordes (los cuales se muestran en el recuadro púrpura de la Figura 4.15) son

verdes, por lo que agregamos estos a un histograma para la clase 3. Por último, la clase 4 correspondiente al fondo se crea construyendo un histograma con los pixeles que no son de la clase 2 o 3.



(a) **(b)**

(c) **(d)**

Figura 4.16: Entrenamiento para la clase 0 y 1.

Para realizar el entrenamiento de la clase 0 se toma una muestra de imagen donde se vean las pelotas y depósito color rojo, como se muestra en la Figura 4.16 (a), después se lleva a cabo una segmentación utilizando las clases 2, 3 y 4 de manera que los pixeles cuya probabilidad de pertenencia a estas clases es 0 son asignados a la clase 0, ya que si su probabilidad es 0 podemos concluir que no pertenecen a la clase 2, 3 o 4, por último se construye un histograma con los pixeles correspondientes a la imagen para esta clase, en la Figura 4.16 (b) se muestra el resultado de la segmentación realizada para obtener este histograma. Para la clase 1 se sigue el mismo procedimiento, esta vez con una muestra de imagen donde se vean las pelotas y depósito azules, en la Figura 4.16 (c) y (d) se muestra la imagen utilizada y su resultado. Para lograr un múltiple entrenamiento, este procedimiento

se puede repetir tantas veces como se desee.

4.8. Procesamiento en paralelo

Para llevar a cabo el procesamiento del robot en tiempo real de manera eficiente, se recomienda aplicar programación paralela, la cual nos permitirá aprovechar el hardware multi-núcleo del Odroid permitiendo que el procesamiento se realize más rápido [Garnica16]. En esta tesis seguiremos el mismo enfoque propuesto por este artículo, el cual consiste en dividir el procesamiento en 3 hilos, los cuales son explicados a continación.

4.8.1. Hilo de la cámara

Este hilo administra la cámara del robot, se encarga de capturar imágenes del ambiente con una resolución de 640×480 pixeles a una velocidad de 24 frames por segundo. El hilo obtiene la última imagen capturada por la cámara y la almacena en una variable global, la cual será accedida por los demás hilos. En el **Algoritmo 4** se muestra este hilo.

Algoritmo 4: Hilo de la cámara.

```
Iniciar la cámara.
salir = falso
mientras salir = falso hacer
    Tomar una fotografía cada (1/24) segundos.
    Copiar fotografía a una variable global.
fin
Liberar variables del hilo y terminarlo.
```

4.8.2. Hilo del procesamiento de colores

Este hilo se encarga de llevar a cabo la segmentación para las 5 clases mencionadas en la sección anterior, se separa cada una de estas clases en una máscara binaria de color. Cada máscara es una imagen binaria con los valores $\{0, 255\}$ donde 255 representa los pixeles que pertenecen a la clase y 0 representa aquellos pixeles que no pertenecen. Posteriormente se aplica el algoritmo para encontrar los cúmulos en la máscara de color azul y rojo, se aplica la transformación de vista de pájaro con una matriz de homografía previamente calculada

para calcular la distancia y ángulo de las pelotas y depósitos. Por último se eliminan los objetos que están más allá del tablero.

Los datos generados por este hilo son compartidos con el hilo del robot para que este pueda tomar decisiones como ir por una pelota o un depósito. En el **Algoritmo 5** se muestra este hilo.

Algoritmo 5: Hilo del procesamiento de colores.

Esperar la primera foto capturada por el hilo de la cámara.

salir = falso

mientras *salir = falso* **hacer**

Copiar imagen global a una variable local.

Calcular la matriz q para 5 clases con el Algoritmo 1.

Crear una máscara de color para cada clase dada por q.

Detectar los cúmulos azules y rojos con el Algoritmo 2.

Diferenciar las pelotas y depósitos de los cúmulos detectados.

Calcular la matriz de homografía y la distancia para cada uno de los objetos utilizando la vista de pájaro.

fin

Liberar variables del hilo y terminarlo.

4.8.3. Hilo para el control del robot

Este hilo se encarga de llevar a cabo la toma de decisiones del robot de servicio, utiliza los datos obtenidos por el hilo de la cámara y procesamiento de colores, este hilo se muestra en el **Algoritmo 6**. Se definen 3 estados posibles, los cuales son los siguientes:

- **Buscar_Pelota:** Consiste en buscar una pelota cercana, en caso de que la cámara del robot vea una pelota ya sea roja o azul, este avanza hacia la pelota más cercana de manera que las pinzas del robot se encuentren a una distancia cercana de la misma (aproximadamente 12 cm), después avanza los 12 cm y toma la pelota con las pinzas. Por último, cambia al estado **Verificar_Pelota**.

Algoritmo 6: Hilo para el control del robot.

```
estado = Buscar_Pelota.
salir = falso
mientras salir = falso hacer
    si estado = Buscar_Pelota entonces
        Moverse hacia la pelota más cercana.
        Calcular distancia d y ángulo θ a la pelota más cercana.
        si 8cm ≤ d ≤ 12cm ∧ −5 ≤ θ ≤ 5 entonces
            Avanzar 12cm y tomar la pelota.
            estado = Verificar_Pelota.
        fin
    fin
    si Estado = Verificar_Pelota entonces
        Levantar la pinza y verificar si se tiene la pelota.
        si Se tiene la pelota entonces
            estado = Buscar_Depósito.
        fin
        en otro caso
            estado = Buscar_Pelota.
        fin
    fin
    si Estado = Buscar_Depósito entonces
        Buscar el depósito del color de la pelota que tiene la pinza.
        Calcular distancia d y ángulo θ del depósito.
        si 8cm ≤ d ≤ 12cm ∧ −5 ≤ θ ≤ 5 entonces
            Avanzar 12cm y soltar la pelota.
            estado = Buscar_Pelota.
        fin
    fin
fin
Liberar variables del hilo y terminarlo.
```

- **Verificar_Pelota:** Consiste en verificar si el robot tomó exitosamente la pelota, para ello, el robot levanta la pinza para acercar la pelota a la cámara y verifica que se tenga la pelota, si esto es cierto, cambia de estado a **Buscar_Depósito**, en caso contrario cambia de estado a **Buscar_Pelota**.

- **Buscar_Depósito:** Consiste en buscar el depósito correspondiente a la pelota que tiene el robot, cuando el depósito está a la vista de la cámara del robot, este avanza

hasta que el depósito se encuentre a una distancia cercana de las pinzas del robot (aproximadamente 12 cm), posteriormente, el robot avanza 12 cm y suelta la pelota. Por último, cambia al estado **Buscar_Pelota**.

4.9. Conclusiones

Las especificaciones técnicas del robot así como los algoritmos aplicados para el mismo fueron presentados, se mostraron los métodos y técnicas utilizadas para que el robot obtenga información del ambiente real en el que trabaja, como lo es la vista de pájaro y la detección de cúmulos. Por último, se mostraron los hilos implementados en el robot donde se aplica la inteligencia y la toma de decisiones del mismo.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos para la segmentación de color, así como el desempeño del robot. Estos resultados fueron comparados con aquellos obtenidos por Garnica et al. [Garnica16]. Los resultados de la segmentación fueron validados utilizando índices de Tanimoto, Ecuación (2.5). Los resultados del desempeño del robot fueron validados midiendo el tiempo que tardaba el robot en terminar la tarea para recoger las pelotas del tablero.

5.1. Resultados de segmentación con pruebas sintéticas

En estos experimentos se aplicó segmentación de color a la imagen de la Figura 2.7 (a). Se aplicó la segmentación utilizando el método de Garnica et al. [Garnica16] para 3 clases y este fue comparado con la segmentación realizada con el método SMDF extendido, dado que el método propuesto en [Garnica16] no maneja coherencia espacial, se hicieron comparaciones con SMDF extendido sin coherencia espacial y SMDF extendido con coherencia espacial. Esta comparación se muestra en la Figura 5.1.

En la Figura 5.1 (a) se muestra la imagen original, en (b) se muestra la segmentación realizada con [Garnica16], en (c) se muestra la segmentación con el método SMDF extendido sin coherencia espacial, y en (d) se muestra la segmentación con el método SMDF extendido con coherencia espacial. Para las tres segmentaciones, la clase 1 se muestra en color azul, la clase 2 en amarillo y la clase 3 en rojo. Para verificar la precisión de cada

segmentación, se utilizáron los índices de Tanimoto [Tanimoto05], en la Tabla 5.1 se muestra la precisión de ambas segmentaciones en términos de porcentaje, también se muestra el promedio de estas segmentaciones.

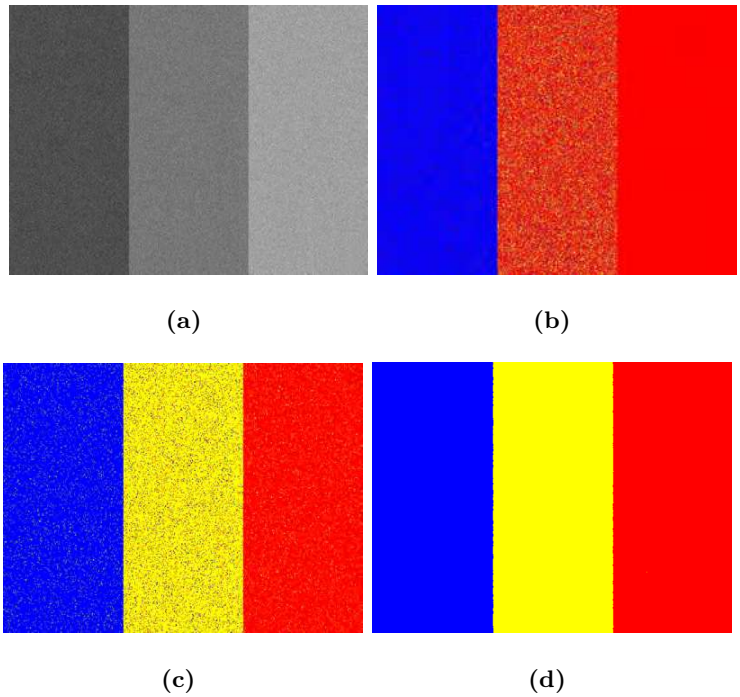


Figura 5.1: Segmentación de color para 3 clases. En (a) se muestra la imagen original, en (b) la segmentación realizada con [Garnica16], en (c) se muestra la segmentación realizada con SMDF extendido sin coherencia espacial, y en (d) la segmentación realizada con SMDF extendido con coherencia espacial.

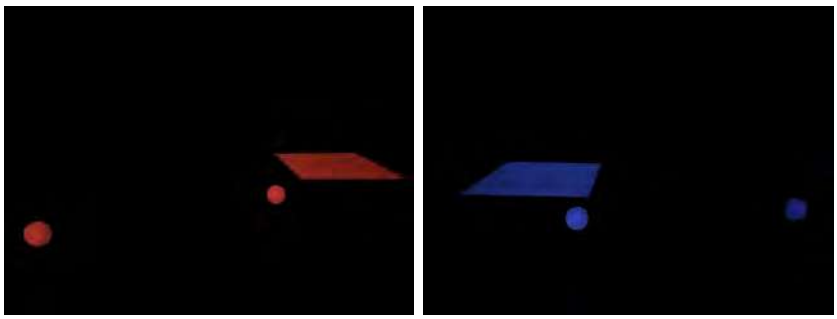
Clase	[Garnica16]	SMDF extendido (sin coherencia espacial)	SMDF extendido (con coherencia espacial)
1	90.58 %	91.15 %	98.92 %
2	21.53 %	82.13 %	97.98 %
3	56.62 %	90.67 %	99.04 %
Promedio	**56.24 %**	**87.98 %**	**98.64 %**

Tabla 5.1: Comparación de segmentación de cada clase para los resultados de la Figura 5.1.

5.2. Resultados de segmentación con pruebas reales

En estos experimentos se aplicó la segmentación de color en imágenes del ambiente real del robot. Esta segmentación fue aplicada utilizando el método propuesto por Garnica et al. [Garnica16], posteriormente se aplicó la segmentación utilizando el método SMDF extendido. La segmentación fue aplicada para la clase 0 que corresponde al color rojo, y la clase 1 que corresponde al color azul, posteriormente, la segmentación para cada clase fue colocada en una máscara binaria.

Para verificar la precisión de cada segmentación, se utilizaron los índices de Tanimoto [Tanimoto05], donde se comparó cada máscara binaria de color con una imagen de validación. Para construir la imagen de validación, se tomó la imagen para la cual se aplicó la segmentación, posteriormente, con un editor de imágenes se seleccionaron los pixeles correspondientes al color que se está verificando, esta selección se hizo con el ojo humano y con la mayor precisión posible. En la Figura 5.2 se muestran 2 imágenes utilizadas para validar la segmentación, donde (a) corresponde a la clase 0, y (b) a la clase 1. Las imágenes mostradas en la Figura 5.2 (a) y (b) fueron utilizadas para validar la imagen de la Figura 5.3 (a).

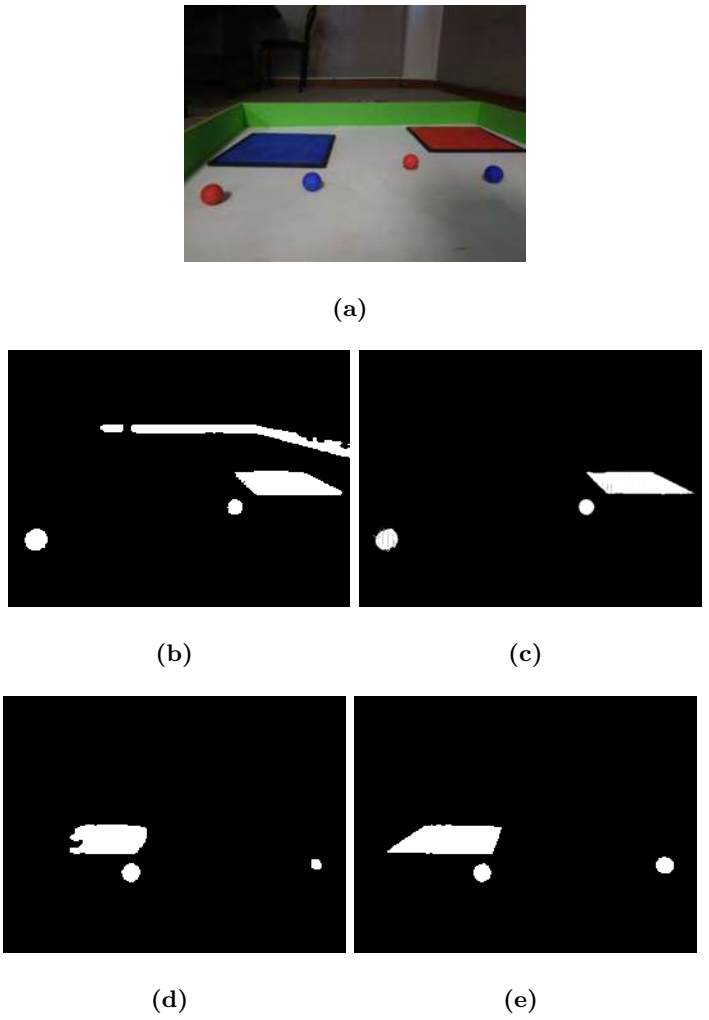


(a) **(b)**

Figura 5.2: Ejemplo de imágenes usadas para la validación.

Se realizaron un total de 18 experimentos con diferentes tipos de iluminación. De estos experimentos se presentan 4, donde seleccionamos el experimento con el mejor desempeño del método SMDF extendido, y también el de su peor desempeño. Los resultados

son mostrados en las Figuras 5.3, 5.4, 5.5 y 5.6. Para las 4 figuras, (a) corresponde a la imagen original, (b) corresponde a la segmentación para la clase 0 con el método del artículo [Garnica16] mientras que (c) corresponde a la segmentación para la misma clase con el método SMDF extendido, (d) corresponde a la segmentación para la clase 1 con el método del artículo [Garnica16] mientras que (e) corresponde a la segmentación para la misma clase con el método SMDF extendido con coherencia espacial.

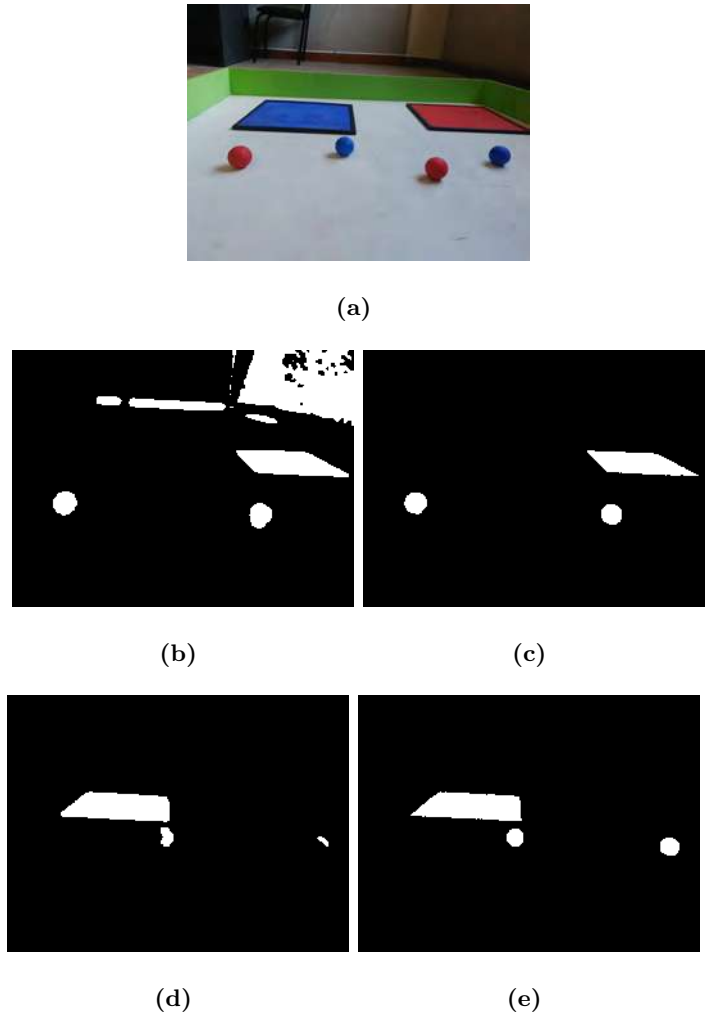


(a) **(b)** **(c)** **(d)** **(e)**

Figura 5.3: Resultados de segmentación para el experimento 1. En (a) se muestra la imagen original, en (b) y (d) se muestran los resultados obtenidos con [Garnica16], en (c) y (e) se muestran los resultados obtenidos con SMDF extendido.

Los experimentos 1 y 2 fueron llevados a cabo en el interior de un edificio. La luz

dentro del edificio varia poco, sin embargo, para realizar pruebas con cambios de iluminación dentro del edifio, el experimento 1 fue realizado con las ventanas del edificio cerradas, evitando así que entre luz del exterior. El experimento 2 fue realizado con las ventanas abiertas permitiendo que entre luz del exterior. El experimento 1 es mostrado en la Figura 5.3, la comparación de ambas segmentaciones para cada clase así como su promedio es mostrado en la Tabla 5.2.



(a)

(b) **(c)**

(d) **(e)**

Figura 5.4: Resultados de segmentación para el experimento 2. En (a) se muestra la imagen original, en (b) y (d) se muestran los resultados obtenidos con [Garnica16], en (c) y (e) se muestran los resultados obtenidos con SMDF extendido.

Clase	[Garnica16]	SMDF extendido
0	45.27 %	93.28 %
1	73.10 %	95.10 %
Promedio	**59.18 %**	**94.19 %**

Tabla 5.2: Comparación de segmentación de cada clase para la Figura 5.3.

El experimento 2 es mostrado en la Figura 5.4, la comparación de ambas segmentaciones para cada clase así como su promedio es mostrado en la Tabla 5.3.

Clase	[Garnica16]	SMDF extendido
0	24.60 %	95.63 %
1	84.54 %	95.56 %
Promedio	**54.57 %**	**95.59 %**

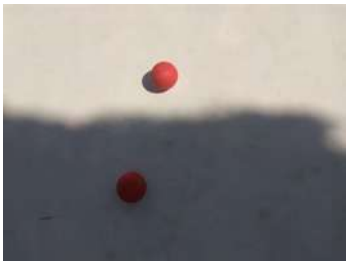
Tabla 5.3: Comparación de segmentación de cada clase para la Figura 5.4.

Los experimentos 3 y 4 fueron realizados en otro ambiente con iluminación diferente, se colocó el tablero en un área abierta a la luz del sol, colocando una parte del tablero en una zona con sombra, de manera que tenemos 2 tipos de iluminación diferentes en una misma imagen, la zona con sombra y la zona con luz solar.

El experimento 3 es mostrado en la Figura 5.5, en este experimento, solo se realizó segmentación para 1 clase, los resultados obtenidos por [Garnica16] se muestran en la Figura 5.5 (b), los resultados obtenidos con SMDF extendido se muestran en (c). La comparación de ambas segmentaciones para cada clase es mostrada en la Tabla 5.4.

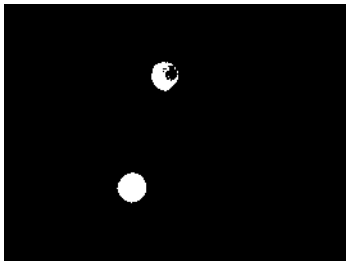
Clase	[Garnica16]	SMDF extendido
0	89.08 %	85.72 %

Tabla 5.4: Comparación de segmentación para la Figura 5.5.



(a)

(b)



(c)

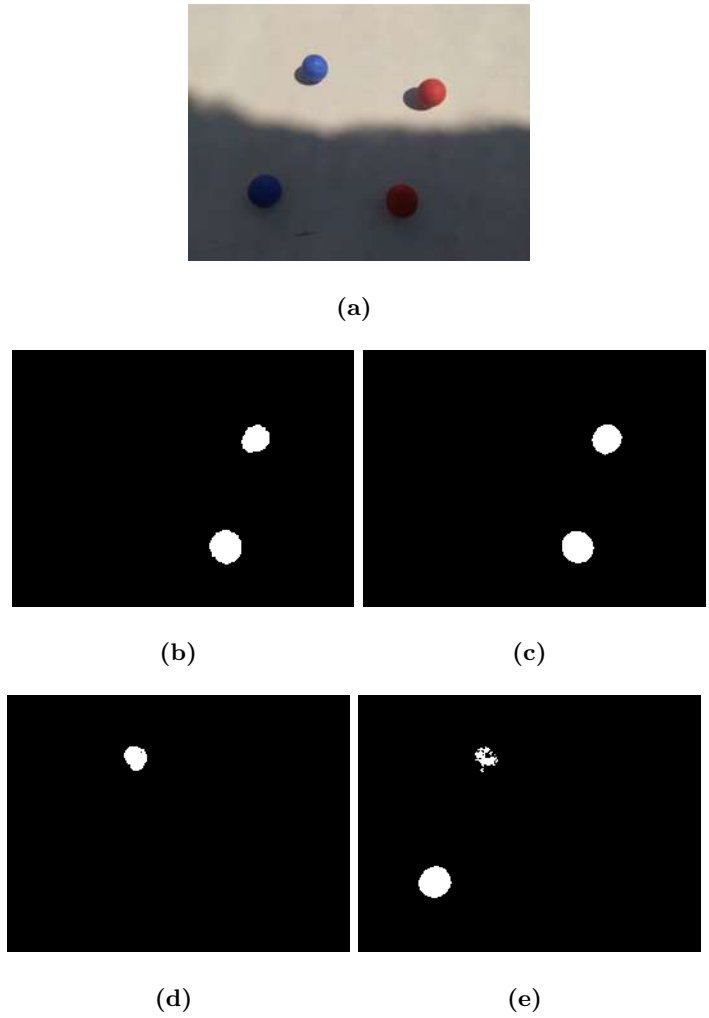
Figura 5.5: Resultados de segmentación para el experimento 3. En (a) se muestra la imagen original, en (b) se muestran los resultados obtenidos con [Garnica16], en (c) se muestran los resultados obtenidos con SMDF extendido.

Se puede observar en este experimento que la segmentación realizada por [Garnica16] dio mejores resultados. Esto es debido a que el entrenamiento aplicado en el método SMDF extendido no fue de buena calidad, para mejorar estos resultados, se puede proponer entrenar con más imágenes.

El experimento 4 es mostrado en la Figura 5.6, la comparación de ambas segmentaciones para cada clase así como su promedio es mostrado en la Tabla 5.5.

Clase	[Garnica16]	SMDF extendido
0	89.66 %	93.85 %
1	35.55 %	76.84 %
Promedio	**62.61 %**	**85.35 %**

Tabla 5.5: Comparación de segmentación de cada clase para la Figura 5.6.



(a) **(b)** **(c)** **(d)** **(e)**

Figura 5.6: Resultados de segmentación para el experimento 4. En (a) se muestra la imagen original, en (b) y (d) se muestran los resultados obtenidos con [Garnica16], en (c) y (e) se muestran los resultados obtenidos con SMDF extendido.

5.3. Resultados de experimentos realizados con el robot

A continuación se presentan los resultados de experimentos del robot recogiendo pelotas. Se realizaron experimentos utilizando los métodos propuestos por el artículo [Garnica16], posteriormente, se realizaron los mismos experimentos utilizando la segmentación SMDF extendido y los algoritmos del robot propuestos en esta tesis. Dado que el tiempo es un factor que determina el ganador de la competencia de robots de servicio, éste será

medido para determinar el desempeño del robot.

En los siguientes experimentos se colocó el robot en una posición fija en el tablero, al mismo tiempo, los depósitos y pelotas fueron colocadas en posiciones seleccionadas dentro del tablero de manera fija, se inició el robot y se esperó a que éste recogiera todas las pelotas y las colocara en sus respectivos depósitos. Para verificar el desempeño del robot, se midió el tiempo que tardó el robot en tomar cada pelota, y el tiempo que tardó en llevarla a su depósito, por último, se midió el tiempo que tardó el robot en terminar su tarea. Cada experimento fue repetido 5 veces para cada método, esto se hizo procurando que los cambios en el ambiente para cada experimento fueran los mínimos posibles.

En estos experimentos el tiempo que tarda el robot en terminar la tarea está determinado por la velocidad y precisión con la que el robot toma la pelota, así como el orden de las pelotas que el robot decide tomar. Dado que los métodos SMDF extendido y [Garnica16] deciden ir por la pelota más cercana que se muestra en la cámara, el orden en el que deciden recoger las pelotas es el mismo.

El experimento 1 fue realizado con el ambiente mostrado en la Figura 5.7, este experimento fue realizado con 1 pelota roja y 1 pelota azul. Se realizaron 5 repeticiones para este experimento. Para la primera prueba, los tiempos (en segundos) de las acciones tomadas por el robot para cada método son mostrados en la Tabla 5.6.

Acción	[Garnica16]	SMDF extendido
Recoger pelota #1	16.62 s	6.87 s
Dejar pelota #1 en depósito	11.28 s	9.63 s
Recoger pelota #2	43.38 s	20.31 s
Dejar pelota #2 en depósito	16.1	18.19 s
Tiempo total	90.3 s	57.13 s

Tabla 5.6: Comparación de desempeño del robot para el experimento 1.

Para las 4 pruebas restantes del experimento 1, en la Tabla 5.7 se muestran los tiempos totales del robot para ambos métodos, al final se muestra el promedio de las 5

pruebas.

Prueba	[Garnica16]	SMDF extendido
1	90.3 s	57.13 s
2	75.61 s	57.8 s
3	78.38 s	60.28 s
4	101.9 s	56.8 s
5	95.22 s	57.45 s
promedio	88.26 s	57.89 s

Tabla 5.7: Comparación del tiempo total de las 5 pruebas del robot para el experimento 1.

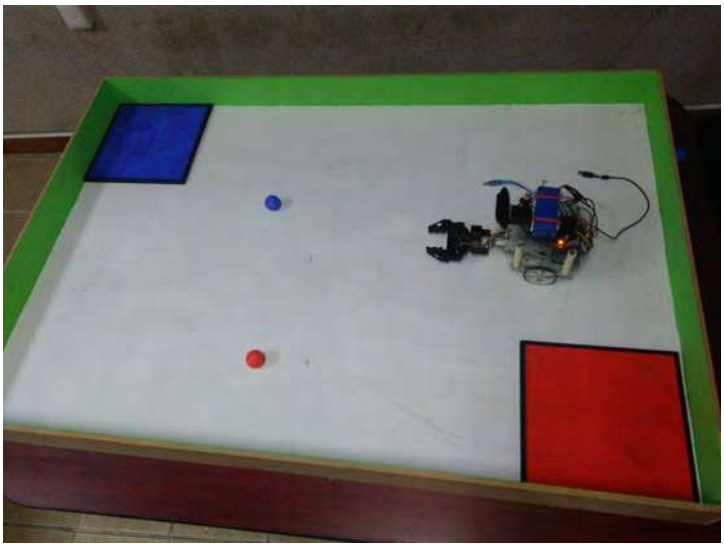


Figura 5.7: Ambiente para el experimento 1.

El experimento 2 fue realizado con el ambiente mostrado en la Figura 5.8, este experimento fue realizado con 2 pelotas rojas y 2 pelotas azules, se realizaron 5 pruebas para este experimento. Para la primera prueba, los tiempos de las acciones tomadas por el robot para cada método son mostrados en la Tabla 5.8.

Acción	[Garnica16]	SMDF extendido
Recoger pelota #1	11.23 s	5.61 s
Dejar pelota #1 en depósito	15.28 s	10.32 s
Recoger pelota #2	31.57 s	21.48 s
Dejar pelota #2 en depósito	15.29 s	19.44 s
Recoger pelota #3	28.89 s	19.66 s
Dejar pelota #3 en depósito	26.71 s	14.3 s
Recoger pelota #4	36.21 s	37.68 s
Dejar pelota #4 en depósito	25.3 s	21.55 s
Tiempo total	192.32 s	152.41 s

Tabla 5.8: Comparación de desempeño del robot para el experimento 2.

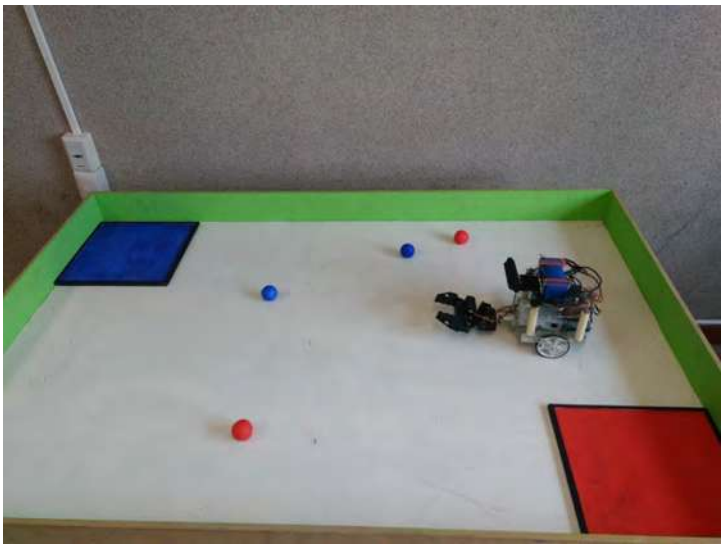


Figura 5.8: Ambiente para el experimento 2.

Para las 4 pruebas restantes del experimento 2, en la Tabla 5.9 se muestran los tiempos totales del robot para ambos métodos, al final se muestra el promedio de las 5 pruebas.

Prueba	[Garnica16]	SMDF extendido
1	192.32 s	152.41 s
2	233.47 s	153.8 s
3	244.32 s	171.16 s
4	203.68 s	160.67 s
5	217.41 s	157.92 s
promedio	218.24 s	159.19 s

Tabla 5.9: Comparación de desempeño de las 5 pruebas del robot para el experimento 2.

5.4. Conclusiones

Se presentaron resultados de la segmentación con el método SMDF extendido comparandola con la segmentación realizada en el artículo [Garnica16]. Se hicieron experimentos de segmentación con una imagen sintética y 4 imágenes del ambiente real del robot y se encontró que el método SMDF extendido es mejor en los dos casos.

Por otro lado se realizaron experimentos con el robot, donde el desempeño de este fue medido en base al tiempo que tardaba en realizar una tarea, como recoger una pelota o dejarla en su depósito. Se encontró que la implementación del robot con las propuestas realizadas en esta tesis es mejor que los algoritmos propuestos en [Garnica16].

Capítulo 6

Conclusiones

En esta tesis se presentó un método de segmentación robusta ante los cambios de iluminación, implementadolo en un robot de servicio de manera que este pueda realizar una tarea específica, los objetivos presentados en esta tesis fueron cumplidos en su totalidad.

6.1. Logros alcanzados

A continuación se presentan los logros alcanzados para esta tesis:

- Se logró llevar a cabo la segmentación de color tolerante a los cambios de iluminación en el ambiente. Se propuso el método SMDF extendido y la segmentación de este método fue comparada con la segmentación realizada en el artículo [Garnica16], donde se demostró que SMDF extendido es mejor.

- La segmentación de color realizada con el método SMDF extendido puede ser llevada a cabo en centésimas de segundos para una imagen con un tamaño de 640×480 pixeles, permitiéndole a esta segmentación ser realizada en tiempo real y aplicada en un robot de servicio como el de esta tesis.

- Se implementaron algoritmos como la vista de pájaro la cual le permite al robot obtener información del ambiente, como lo es la distancia y ángulo de las pelotas y depósitos, la eficacia de este algoritmo fue demostrada al realizar los experimentos con el robot.

- Se presentó el algoritmo `Encontrar_Cúmulos` el cual permite al robot detectar las pelotas y depósitos así como diferenciaros. Su eficacia se demostró en los experimentos realizados con el robot.

- Por último se presentó el desempeño del robot en su ambiente real, donde debía realizar la tarea de recoger las pelotas de color y dejarlas en sus respectivos depósitos. El robot fue capaz de realizar esta tarea de manera exitosa y se comparó su desempeño con la implementación del robot realizada en el artículo [Garnica16], donde se demostró que los algoritmos presentados en esta tesis son mejores.

6.2. Trabajo futuro

El trabajo futuro que se puede desarrollar para complementar este trabajo de tesis es el siguiente:

- El método de segmentación SMDF extendido utiliza la información de color de los pixeles de la imagen en el espacio de color $[R, G, B]$. Se propone como trabajo futuro implementar el mismo algoritmo utilizando el espacio de color $[H, S, V]$ el cual contiene mejor distribuida la información de color con respecto al espacio $[R, G, B]$.

- Se propone implementar mejoras para la inteligencia del robot. Una de estas es hacer un mapa de la ubicación del robot sobre su ambiente, de esta manera, cuando el robot avance o rote, tendrá conocimiento de en que parte del tablero se encuentra, así como los depósitos y pelotas ubicadas a su alrededor.

- Se propone mejorar el modelo de detección de cúmulos, de tal manera que este pueda identificar pelotas que se encuentran sobrepuestas entre ellas y los depósitos.

- Se propone desarrollar un modelo que permita al robot realizar la tarea sin que exista la restricción de que el piso sea plano.

- Se propone desarrollar un modelo de segmentación que que funcione con objetos cotidianos (por ejemplo, una pelota de tenis o golf).

Referencias

[Adams94] Adams, R. y Bischof, L. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

[Akram13] Akram, A. y Ismail, A. Comparison of edge detectors. *Int. J. Comp. Sci. Information Technol. Res*, 1:16–24, 2013.

[Bishop06] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0387-31073-2.

[Calderon15] Calderon, F., Flores, J., y Garnica-Carrillo, A. A fast algorithm for binary segmentation using color information. *En Power, Electronics and Computing (ROPEC), 2015 IEEE International Autumn Meeting on*, págs. 1–7. IEEE, 2015.

[com16] Competencia de robots de servicio. Competencia de robots de servicio, http://dep.fie.umich.mx/ lromero/6EncuentroRobotica/anexos/lineamientoscompetencias/competenciaderobotsdeservicio.pdf, 2016. Consulta: Junio 2018.

[Dempster77] Dempster, A. P., Laird, N. M., y Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, págs. 1–38, 1977.

[Duda73] Duda, R. O., Hart, P. E., Stork, D. G., et al. *Pattern classification*, tomo 2. Wiley New York, 1973.

[Forsyth11] Forsyth, D. A. y Ponce, J. *Computer Vision: A modern approach*. Pearson, 2ª edón., 2011. ISBN 978-0136085928.

[Freund00] Freund, J., Miller, I., y Marylees, M. *Estadistica matematica con aplicaciones*. Prentice Hall, 6ª edón., 2000. ISBN 970-17-0389-8.

[Garnica16] Garnica, A. C., Gómez, S. D., Gómez, J. D., y Romero, L. M. Development of a mobile service robot for classify objects and place them in containers. *En Power, Electronics and Computing (ROPEC), 2016 IEEE International Autumn Meeting on*, págs. 1–6. IEEE, 2016.

[Gonzalez09] Gonzalez, A. J. *Técnicas de percepción activa para seguimiento de objetos mediante robots en entornos urbanos*. Proyecto Fin de Carrera, Universidad de Sevilla, 2009.

[Iocchi06] Iocchi, L. Robust color segmentation through adaptive color distribution transformation. *En Robot Soccer World Cup*, págs. 287–295. Springer, 2006.

[Kaehler16] Kaehler, A. y G., B. *Learning OpenCV 3*. O'Reilly, 2016. ISBN 978-1-491-93799-0.

[Khan14] Khan, J. F. y Bhuiyan, S. M. Weighted entropy for segmentation evaluation. *Optics & Laser Technology*, 57:236–242, 2014.

[Mumford85] Mumford, D. y Shah, J. Boundary detection by minimizing functionals. *En IEEE Conference on Computer Vision and Pattern Recognition*, tomo 17, págs. 137–154. San Francisco, 1985.

[odr18] Odroid-u3 specifications. Odroid-U3 Hardkernel, https://www.hardkernel.com/main/products/prdtinfo.php?gcode=g138745696275tabidx=1, 2018. Consulta: Junio 2018.

[ope18] Camera calibration with opencv. Camera calibration with OpenCV, https://docs.opencv.org/2.4/doc/tutorials/calib3d/ cameracalibration/cameracalibration.html?, 2018. Consulta: Julio 2018.

[Pearson94] Pearson, K. Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical transactions of the Royal Society of London*, 186(Part I):343–424, 1894.

[Prince12] Prince, S. J. *Computer vision: Models, learning and inference*. Cambridge University Press, 2012.

[Pukelsheim94] Pukelsheim, F. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[Romero16] Romero, L. M. *Visión computacional, un enfoque bayesiano*, 2016. Notas de clase.

[Tanimoto05] Tanimoto, P.-N., Steinbach, M., y Kumar, V. *Introduction to data mining*, tomo 2. 1ª ed$^{\underline{\text{ón}}}$., 2005.

[Viola01] Viola, P. y Jones, M. Rapid object detection using a boosted cascade of simple features. *En Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, tomo 1. IEEE, 2001.

[Wasik02] Wasik, Z. y Saffiotti, A. Robust color segmentation for the robocup domain. *Proc. of the Int. Conf. on Pattern Recognition*, 1, August 2002.

[Yuheng17] Yuheng, S. y Hao, Y. Image segmentation algorithms overview. *arXiv preprint arXiv:1707.02051*, 2017.