BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN INGENIERÍA ELECTRÓNICA, OPCIÓN INSTRUMENTACIÓN ELECTRÓNICA

Tesis para obtener el grado de

MAESTRO EN INGENIERÍA ELECTRÓNICA

"Algoritmo para la Detección de Lugares Disponibles en un Estacionamiento"

Presenta:

Víctor Romero Bautista*

Asesores:

Dr. Aldrin Barreto Flores

Dr. Salvador E. Ayala Raggi

*Becario CONACYT

10 de diciembre del 2021

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca recibida para la realización de mis estudios de maestría.

A la Benemérita Universidad Autónoma de Puebla (BUAP) que me condonó la colegiatura durante mis estudios de posgrado en la Maestría en Ingeniería Electrónica opción Instrumentación Electrónica.

A la Facultad de Ciencias de la Electrónica (FCE) por el apoyo infraestructural tanto en el uso de laboratorios como en plataformas digitales que permitieron llevar a cabo de manera adecuada mis estudios en la Maestría en Ingeniería Electrónica opción Instrumentación Electrónica.

A mis asesores de tesis, el Dr. Aldrin Barreto Flores y el Dr. Salvador Eugenio Ayala Raggi por haberme brindado su apoyo y valiosa dirección en este trabajo.

A los miembros de mi jurado, M.C. Ana María Rodríguez Domínguez, Dra. María Monserrat Morín Castillo y M.C. Selene Edith Maya Rueda, por aportar sus valiosas recomendaciones durante el desarrollo de este trabajo.

A la coordinadora de la Maestría en Ingeniería Electrónica opción Instrumentación Electrónica, M.C. Ana María Rodríguez Domínguez por su dedicación y apoyo.

Al cuerpo docente de la Maestría en Ingeniería Electrónica opción Instrumentación Electrónica por su importante aportación en mi formación académica.

A mis padres, hermanas y hermano por apoyarme constantemente en mis decisiones.

A mi novia Cecilia por brindarme su apoyo incondicional.

A mis compañeros de maestría por los momentos y experiencias compartidas.

Víctor Romero Bautista

Índice General

1. Introducción **3**

1.1. Trabajos en el estado del arte 4

1.2. Objetivos 7

1.3. Justificación 7

1.4. Descripción del trabajo propuesto 8

2. Marco Teórico **11**

2.1. Visión por computadora 11

2.2. Las redes neuronales pulso-acopladas (PCNN) 19

2.3. Aprendizaje máquina 24

2.4. La tarjeta Raspberry Pi 4 29

3. Diseño e implementación del algoritmo **31**

3.1. Imágenes adquiridas de un estacionamiento 31

3.2. Segmentación de las imágenes 32

3.2.1. Método de segmentación por zonas 33

3.2.2. Método de segmentación basado en suma de pulsos de máxima entropía . . 34

3.2.3. Método de segmentación basado en pulsos de máxima entropía 37

3.2.4. Método basado en regiones de interés (ROI) 40

3.2.5. Otros métodos de segmentación explorados 42

3.3. Extracción de características 46

3.3.1. Método basado en la obtención de firma $G[n]$ 46

3.3.2. Método basado en matriz de co-ocurrencia (GLCM) 48

3.3.3. Método basado en la obtención de $G[n]$ y PCA 49

3.4. Identificación 52

3.4.1. Método basado en $G[n]$ y red neuronal perceptrón multicapa 52

3.4.2. Método basado en GLCM, área, color y MLP 53

3.4.3. Método basado en la $G[n]$, PCA y MLP 54

3.4.4. Método basado en el histograma y KNN 55
3.5. Algoritmo general 58
3.6. Interfaz de usuario 63
3.7. Implementación en Raspberry Pi 67

4. Resultados **71**
4.1. Resultados de algoritmo 3.5.1 71
4.2. Resultados de algoritmo 3.5.2 73
4.3. Resultados de algoritmo 3.5.3 74
4.4. Resultados de algoritmo 3.5.4 75
4.5. Resultados de algoritmo 3.5.5 76
4.6. Resultados del algoritmo general 77
4.7. Resultados de implementación del algoritmo en Raspberry Pi 79

5. Conclusiones **81**
5.1. Trabajo futuro 83

A. Publicaciones **3**
A.1. Participación en el Congreso Internacional de Innovación Tecnológica y Computación (CIITEC 2020) 3
A.2. Participación en el 8vo. Congreso Internacional de Robótica y Computación (CIRC 2021) 4
A.3. Participación en la XVII Semana Nacional de Ingeniería Electrónica y II Semana Iberoamericana de Ingeniería Electrónica (SENIE 2021) 4
A.4. Participación en el III Congreso Internacional y XI Congreso Nacional de Ciencias de la Computación (CONACIC 2021) 5

Capítulo 1

Introducción

El sentido de la vista, es uno de los cinco sentidos que el ser humano utiliza día con día para interactuar con el entorno que lo rodea, siendo este sentido sobre el cual dependemos en mayor medida que de los demás (tacto, olfato, oído y gusto), ya que a través de la vista se adquiere constantemente una gran cantidad de información, que nos permite desenvolvernos ya sea para navegación o desempeñar alguna tarea, según se requiera.

A lo largo de los años, principalmente con el desarrollo computacional, se han llevado a cabo diferentes trabajos por parte de la comunidad de investigadores, con el objetivo de emular el comportamiento humano, mediante la disciplina denominada inteligencia artificial o IA. La cual fue fundada en el año de 1956, en una reunión que tuvo una duración de dos meses en el colegio de Dartmouth, algunos de sus fundadores son: Marvin Minsky, Claude Shannon y John McCarty. Esta disciplina está conformada por diferentes sub-áreas, tales como: la robótica, visión por computadora y aprendizaje automático. El desarrollo de trabajos relacionados con la IA aún no alcanza la capacidad de efectividad que puede tener un ser humano [9], sin embargo, en el área de visión por computadora, se han logrado desarrollar sistemas capaces de desempeñar tareas en donde el hombre emplearía su sentido de la vista y su habilidad para el reconocimiento de patrones, como son: la auto-navegación de vehículos, monitoreo del tráfico urbano, identificación de huellas dactilares, verificación de firmas, reconocimiento de caracteres, entre otros.

Una de las principales aplicaciones de los sistemas basados en inteligencia artificial para ambientes urbanos, es la vigilancia e inspección del tráfico. Estos sistemas contribuyen a reducir problemas de contaminación (emisión de gases contaminantes provenientes de los automóviles), reducir accidentes automovilísticos, así como, asistir a los conductores para tomar mejores decisiones durante la navegación. Dentro de estos, uno de los temas de investigación, son los sistemas de asistencia a estacionamientos, los cuales tienen el objetivo de proporcionar información de la disponibilidad de lugares en un estacionamiento. Estos sistemas cada vez desempeñan un papel importante para mejorar el flujo del tráfico en las ciudades, ya que debido al incremento constante de automóviles es común el consumo de tiempo y combustible innecesarios mientras se intenta

localizar un lugar de estacionamiento vacío. Normalmente alrededor del 63 % de los usuarios demoran entre 5 y 10 minutos en encontrar un lugar disponible, el 30 % dilata de 11 a 20 minutos y el restante 7 % prolonga su búsqueda por un tiempo de 25 a 35 minutos [24].

Como resultado se han desarrollado diferentes sistemas de asistencia a estacionamientos, donde se ofrece, guiar al conductor hasta encontrar un lugar vacío. Para estos sistemas se utilizan diferentes tipos de tecnologías, en [25] G. Revathi clasifica a los sistemas de asistencia a estacionamientos e indica la tecnología utilizada para cada uno, donde se destacan los sistemas de información de lugares disponibles en estacionamientos. Los cuales se dividen en sistemas basados en sensores de presencia, mismo que son colocados de manera estratégica, en cada espacio de estacionamiento que se requiera monitorear, de esta manera los sensores emitirán una señal cuando el espacio de estacionamiento se encuentre vacío y otra distinta para indicar cuando esté ocupado. Por otro lado, se encuentran los sistemas basados en visión por computadora, los cuales proponen paradigmas para la detección de lugares disponibles, empleando cámaras digitales y técnicas de visión artificial.

Algunos retos de los sistemas basados en visión artificial son expuestos en [15], donde C. Huang menciona a los cambios luminosidad que suelen ocurrir durante el día, los efectos de sombra y oclusión entre objetos. Para superar estos retos se han propuesto diferentes métodos. Algunos de éstos son: los métodos basados en el mapeo del estacionamiento para la generación de regiones de interés (ROI), métodos que emplean la resta del fondo, o bien, métodos orientados a la detección de líneas y esquinas, algunas técnicas recientes, emplean el uso de las redes neuronales convolucionales (CNN).

1.1. Trabajos en el estado del arte

Algunos de los trabajos encontrados en la literatura son presentados en la tabla 1.1, en donde se presenta el método empleado por el autor, y una comparación entre las ventajas e inconvenientes que se suelen presentar en cada uno de éstos.

Los autores C. Huang [15] y P. Tatulea [30], utilizaron un método basado en el mapeo del estacionamiento, que consiste en ubicar de manera estratégica, las regiones donde se úbican los espacios de estacionamiento que se deseen monitorear. Ésto lo hacen, indicando manualmente las coordenadas de cada región de interés (ROI), dentro de la imagen de captura. P. Tatulea, además de ubicar las ROI, para el reconocimiento de éstas, extrae las siguientes características: el histograma de gradientes orientados (HOG), detector de esquinas mediante la transformada de características invariantes (SIFT) y valores de tonalidad de color en HSV, YUV y YCrCb. Por otro lado, C. Huang combina el mapeo del estacionamiento, con la implementación de sensores infrarrojos, sin embargo, ésto conlleva una mayor inversión económica para el desarrollo del sistema.

Por otro lado, los autores S. Lee [17], J.K. Suhr [29] y L.W. Tsai [31], utilizaron un método denominado automático, basados en la detección de líneas y esquinas, que corresponden a las marcas de delimitan los lugares de estacionamiento. Estos trabajos pueden detectar espacios disponibles de estacionamientos de forma automática en ambientes controlados (lugares donde hay mínimos cambios de iluminación), por lo cual presentan baja efectividad al presentarse poca iluminación, imágenes distorsionadas y oclusión entre objetos.

Los trabajos presentados por B. Y. Cai [6], C. K. Ng [22] y S. Banerjee [4], utilizan métodos basados en redes neuronales artificiales (ANN), donde B. Y. Cai, utiliza redes neuronales convolucionales profundas *DCNN*; el sistema implementa un rastreo de vehículos y es capaz de detectar errores causados por la oclusión de objetos, para lo cual demanda recursos computacionales elevados. En cambio C.K. Ng, utiliza un modelo ligero de redes neuronales convolucionales, para implementarlo en una tarjeta Raspberry Pi 3; este sistema presenta baja eficiencia en ambientes con poca iluminación. S. Banerjee, propone un sistema basado DCNN, este método presenta una efectividad del 72%, donde su funcionalidad disminuye con las distorsiones de luz, que suelen ocurrir durante el día.

En general los sistemas basados en visión por computadora utilizados para la detección de lugares disponibles en estacionamientos, presentan diferentes dificultades al momento de ser implementados, que a su vez afectan en la efectividad de estos sistemas. Particularmente, uno de los problemas más importantes que se suele presentar, es la distorsión ocasionada por los diferentes efectos de luz que suelen ocurrir durante el día, los cuales se deben en su mayoría a factores climáticos, ya que la iluminación varía en un día soleado, nublado o lluvioso.

En este trabajo se propone diseñar un algoritmo basado en inteligencia artificial, que permita llevar a cabo la detección de lugares disponibles en un estacionamiento, basado en métodos de visión por computadora. Para la etapa de segmentación se hace uso de redes neuronales de pulso acopladas (PCNN), en su forma simplificada, el modelo de intersección cortical (ICM). Para la extracción de características se hace uso de la firma ($G[n]$), producida por la misma red ICM. Por último, en la etapa de reconocimiento, se emplea la red neuronal perceptrón multicapa. Además, se lleva a cabo el desarrollo de una interfaz de usuario, la cual indique el porcentaje de lugares disponibles en el estacionamiento.

Autor	Método	Ventaja	Inconveniente
B. Y. Cai [6], 2019	Deep Convolutional Neural Networks DCNN	Implementa un filtro de rastreo de vehículos, que remueve el ruido y detecta fallos causados por la oclusión entre objetos	Utiliza recursos computacionales elevados en hardware para su correcto funcionamiento
C. Huang [15], 2013	Basado en infraestructura	Implementa un método planar basado en un modelo de estacionamiento 3D con superficies completamente planas	Esta adaptado únicamente a la morfología del estacionamiento
H. G. Jung [16], 2014	Semi Automático	Utiliza un algoritmo basado en la detección de líneas y esquinas que forman los espacios de estacionamiento	Designado únicamente para estacionamientos que cumplan con las características de utilizar espacios de estacionamiento rectangulares
S. Lee [17], 2016	Automático	Propone un algoritmo que incluye un paso de validación de espacios de estacionamiento en una integración probabilistíca que tiene mayor flexibilidad a patrones irregulares	Presenta baja efectividad en imágenes con distorsión
C. K Ng [22], 2018	Convolutional Neural Networks	Implementa un modelo ligero de red neuronal convolucional en una tarjeta Raspberry para detectar automóviles mal estacionados	Baja efectividad en ambientes con poca iluminación
J. K. Suhr [29], 2012	Automático	Utiliza un método de aproximación para encontrar espacios de estacionamiento usando secuencias de imágenes AVM	No es efectivo en imágenes con oclusión entre el automóvil y las marcas del espacio de estacionamiento
P. Tatulea [30], 2019	Basado en Infraestructura	Utiliza la ubicación de ROI, y para su reconocimiento el HOG, SIFT, HSV, YUV y YCrCb	Baja efectividad al presentarse sombras
L. W. Tsai [31], 2007	Automático	Utiliza un modelo de transformación de color, el cual puede identificar vehículos en una imagen bajo diferentes tonalidades	Baja efectividad en ambientes con poca iluminación
S. Banerjee [4], 2019	Deep Convolutional Network	Propone un sistema de estacionamiento inteligente basado en una arquitectura de redes convolucionales profundas	Presenta una efectividad baja del 72 %, su funcionalidad disminuye con las distorsiones de luz durante el día

Tabla 1.1: Comparativa de trabajos realizados para la detección de lugares disponibles en estacionamientos.

1.2. Objetivos

Para este trabajo se establecieron los siguientes objetivos.

Objetivo General

Desarrollar un algoritmo basado en inteligencia artificial que permita la detección automática de los lugares disponibles en un estacionamiento durante el día.

Objetivos Específicos

- Generar un conjunto adecuado de imágenes de un estacionamiento.
- Diseñar un algoritmo de segmentación de imágenes utilizando el modelo de redes neuronales pulsantes (PCNN).
- Diseñar un algoritmo para la obtención de características de los espacios disponibles a través del análisis digital de imágenes.
- Diseñar un algoritmo basado en redes neuronales, para identificar espacios disponibles y ocupados en el estacionamiento.
- Crear una interfaz de usuario, donde se indique el porcentaje de espacios libres y ocupados.

1.3. Justificación

Los sistemas de detección de lugares disponibles en un estacionamiento basados en visión por computadora, se enfrentan al reto de tener un sistema capaz de realizar el proceso de detección de los lugares de un estacionamiento cuando, se presentan cambios de luminosidad en las imágenes, oclusión entre objetos y efectos de sombra. Los cambios de luz normalmente son ocasionados por el clima, por ejemplo, en un día nublado, las imágenes que se adquieran del estacionamiento tendrán una iluminación homogenea, en cambio, durante un día soleado las imágenes presentaran mayor cantidad de luz, con la posibilidad de generar sombras. La oclusión entre objetos ocurre, cuando un objeto se superpone sobre otro. Los efectos de sombra aparecen cuando se tienen objetos de gran altura, que a su vez, dependiendo de la hora del día pueden generar sombras, las cuales obstruyen la correcta iluminación de los objetos de interés, un ejemplo de ésto, pueden ser los árboles que comúnmente son colocados en el perímetro de los estacionamientos.

Estos diferentes escenarios que se presentan influyen directamente en la efectividad de los sistemas de detección de lugares disponibles en estacionamientos, ya qué agregan o disminuyen

información en las imágenes del estacionamiento, por lo cual se requiere de uno o mas algoritmos que contribuyan entre sí para disminuir los problemas de reconocimiento de objetos cuando se presentan estos escenarios.

Las contribuciones que se plantean para este trabajo son:

- El diseño de un algoritmo para segmentación de imágenes utilizando redes neuronales pulsantes (PCNN), el cual debido a que conserva propiedades originales de la imagen, puede contribuir a tener mejores resultados en el proceso de detección de los objetos de interés en diferentes condiciones de luz.
- El diseño de un algoritmo para identificación de los lugares de estacionamiento disponibles en un estacionamiento basado en redes neuronales que sea capaz de realizar el proceso de detección en diferentes condiciones del clima durante el día.
- La implementación de los algoritmos diseñados para detección de los lugares disponibles en una tarjeta *Raspberry Pi*.
- El desarrollo de una interfaz de usuario donde se indique el porcentaje de lugares de estacionamiento disponibles para disminuir el tiempo de búsqueda de un lugar de estacionamiento vacío.

1.4. Descripción del trabajo propuesto

En este trabajo se desarrolló un algoritmo basado en inteligencia artificial, que es capaz de detectar lugares disponibles en un estacionamiento, e indicar a los usuarios la ubicación de éstos, para que puedan conducir directamente hacia donde se encuentra el lugar vacío.

Para lograr ésto, propusieron cinco tareas, las cuales se presentan en el diagrama de bloques de la figura 1.1. La primera (inciso a) consistió en recolectar y almacenar imágenes de un estacionamiento, con el fin de generar un banco de información. Como segunda tarea (inciso b), se diseñó un algoritmo de segmentación basado en el modelo de redes neuronales pulsantes (PCNN), que posteriormente se aplicará a las imágenes del banco de información generado. Ésto con el fin de resaltar y separar los objetos de interés. La tercer tarea (inciso c), consistió en el diseñó de un algoritmo para extraer las características de los objetos previamente segmentados. Ésto permitió el reconocimiento de los objetos de interés. Posteriormente, en la tarea cuatro (inciso d), se diseñó un algoritmo basado en redes neuronales artificiales, para identificar los espacios disponibles y ocupados en el estacionamiento. Por último, en la tarea cinco (inciso e), se desarrolló una interfaz de usuario, donde se muestra el porcentaje de los espacios disponibles en el estacionamiento, así

como la ubicación de los mismos, con el objetivo de mostrar a los usuarios opciones de los lugares disponibles para estacionarse.

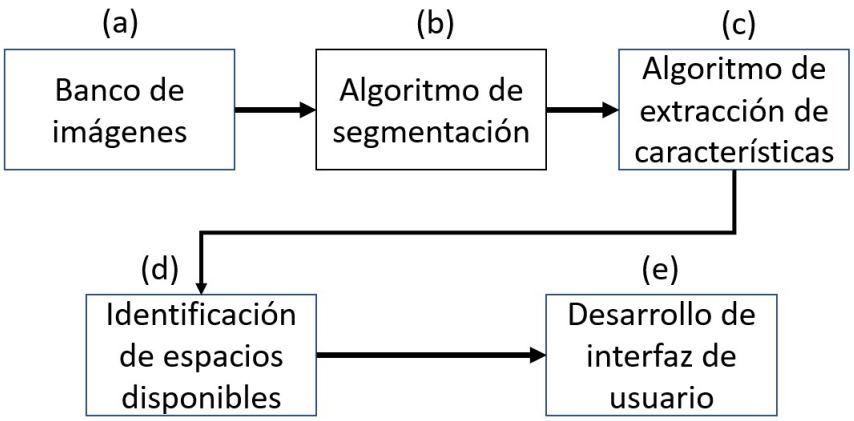
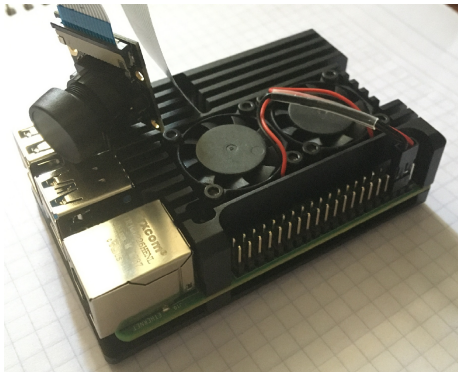


Figura 1.1: Diagrama de bloques del trabajo a realizado.

Los algoritmos propuestos en este trabajo se implementaron en el lenguaje de programación *Python*, y fueron ejecutados en una tarjeta *Raspberry pi* 4 (inciso (a) de la figura 1.2). Esta tarjeta cuenta con un procesador Quad Core Cortex-A72 con velocidad de 1.5 GHz, y presenta modelos de memoria RAM en 2, 4 y 8 GB.



(a)

Figura 1.2: Tarjeta *Raspberry Pi* 4.

Capítulo 2

Marco Teórico

Para cumplir con cada uno de los objetivos planteados en esta tesis, se requiere de un estudio previo de diferentes áreas del conocimiento, tales como: visión por computadora y aprendizaje automático. Cada una con tópicos particulares que permitieron el desarrollo de cada etapa del proyecto. En este capítulo se presentan las bases teóricas en las que se sustenta este trabajo. Se comienza con el área de visión por computadora, donde se presentan algunas técnicas de procesamiento digital de imágenes, posteriormente se abordan las redes neuronales pulso-acopladas (PCNN), utilizadas para la etapa de segmentación, y por último, el área de aprendizaje automático, donde se presentan algoritmos de clasificación basados en aprendizaje supervisado y no supervisado, que serán empleados para la detección de espacios disponibles en el estacionamiento.

2.1. Visión por computadora

El área de visión por computadora también conocida como visión artificial, pertenece al conjunto de sub-áreas del conocimiento que conforman la disciplina de la inteligencia artificial (IA), y se centra en el desarrollo de algoritmos que traten de lograr que una máquina simúle hasta cierto grado el proceso de visión biológico [7]. Una imagen digital contiene información del mundo físico, y es generada mediante un proceso de muestreo que normalmente entrega una imagen a color con tres componentes, que son rojo, verde y azul, o RGB por sus siglas en ingles, donde cada componente, es representado por una matriz, y los elementos dentro de ésta (pixeles), representan los niveles de intensidad para cada punto en la imagen. Para el procesamiento de imágenes se suele trabajar con escala de grises, lo cual elimina las componentes RGB y solo se queda con una componente general que representa el promedio de los niveles de intensidad en RGB.

Representación de una imagen digital

Una imagen digital se puede definir como una función bidimensional que cuantifica la intensidad de luz, y normalmente se representan como una matriz definida por I (ecuación 2.1), donde

los componentes dentro de ésta corresponden a los valores de intensidad, y los sub-índices f, c representan las dimensiones de la imagen I, en filas y columnas respectivamente, es decir que cada componente de la matriz representa a un píxel de la imagen I.

$$I = \begin{pmatrix} i_{1,1} & i_{1,2} & i_{1,c} \\ i_{2,1} & i_{2,2} & i_{2,c} \\ i_{f,1} & i_{f,2} & i_{f,c} \end{pmatrix} \tag{2.1}$$

El histograma de una imagen

El histograma de una imagen describe la frecuencia con la que se presentan los valores de intensidad en la misma, por lo tanto brinda información acerca de la iluminación, contraste y la dinámica. Para una imagen en escala de grises definida por $I_{m,n}$ con intensidades entre 0 a 255, contendrá un histograma H con 256 valores, donde cada valor de H es definido por $h(i)$, que representa la cantidad de píxeles en la imagen con intensidad i, para un i dentro del intervalo $[0-255]$; en la ecuación (2.2) se muestra la representación del histograma, donde $card^2$ indica la cardinalidad del conjunto.

$$h(i) = card^2\{(m,n)|I(m,n) = i\} \tag{2.2}$$

Filtro de la mediana

Este filtro permite eliminar artefactos y pequeñas estructuras no deseadas en la imagen sin afectar significativamente los bordes como en el caso del filtro Gaussiano. La mediana se puede definir como: el valor de la variable que deja el mismo número de datos antes y después que él. Es decir el conjunto de datos menores que la mediana representaran el 50 % de los datos, y los que sean mayores representaran el otro 50 % del total de datos de la muestra [7]. En la siguiente ecuación se presenta la expresión matemática de este filtro:

$$M_e = x_{\frac{n+1}{2}} \tag{2.3}$$

donde $x_1, x_2, x_3 ..., x_n$ son los datos de una muestra ordenada de forma creciente.

El filtro de la mediana sustituye la intensidad original de cada píxel de la imagen, por la mediana de los valores de intensidad dentro de la región de influencia definida por el filtro. Normalmente la dimensión que se suele utilizar para este filtro es de $3x3$ o bien $5x5$ píxeles.

Filtros morfológicos

Los filtros morfológicos son capaces de influir sobre estructuras en una imagen, y son designadas previamente mediante la definición de un elemento estructural. Estos filtros fueron originalmente concebidos para ser implementados sobre imágenes binarias, en donde los valores de intensidad en cada píxel pueden ser 0 o 1. En una imagen binaria normalmente se toma el valor 0 para representar el fondo de la imagen y el 1 para las estructuras de interés. Además, estos filtros también pueden ser implementados sobre imágenes en escala de grises, tomando en cuenta los valores de intensidad máximo y mínimo, en lugar de 1 y 0.

Las operaciones de reducción y crecimiento son dos de las operaciones morfológicas fundamentales, los cuales por la relación cercana que tienen entre los procesos físicos de dilatación y erosión, llevan este nombre [7]. A continuación, son presentados los filtros de dilatación y erosión para imágenes binarias y en escala de grises, comenzando por la definición de la estructura de referencia.

La estructura de referencia

Para la operación de un filtro morfológico se necesita la caracterización de una máscara llamada estructura de referencia. Al igual que una imagen binaria, la estructura de referencia contiene también solo elementos 0 y 1, por lo que se podría definir tal como se muestra en la ecuación (2.4).

$$H(i,j) \in \{0,1\} \tag{2.4}$$

Dilatación

La dilatación es una operación morfológica que permite añadir una capa de píxeles a las estructuras en una imagen. La estructura de referencia permite controlar dicho crecimiento. Esta operación es definida en la ecuación (2.5).

$$I \oplus H = \{(x',y') = (x+i,y+i) | (x',y') \in P_I, (i,j) \in P_H\} \tag{2.5}$$

donde el conjunto de puntos que constituyen a la dilatación de una imagen *I* y la estructura de referencia *H*, queda definido por todas las posibles combinaciones de los pares de coordenadas de los conjuntos de puntos P_I y P_H. Se podría también interpretar a la operación de dilatación como el resultado de añadir los píxeles de valor 1 a la imagen, de acuerdo a la forma correspondiente a la estructura de referencia.

Erosión

La operación de erosión corresponde añadir una capa de píxeles al fondo de la imagen, debido a ésto, el efecto que presenta, es la disminución de la estructura, y esta operación se expresa en la ecuación (2.6).

$$I\Theta H = \{(x',y')|(x'+i,y'+j) \in P_I(i,j), \forall (i,j) \in P_H\} \tag{2.6}$$

donde para cada punto (x',y') de la imagen, el resultado se compone por los puntos (x',y'), para los cuales todos los posibles valores $(x'+i,y'+j)$ se encuentra en I. Este proceso puede ser interpretado como: si un píxel (x',y') resultado de la erosión es uno si la estructura de referencia centrada en este píxel coincide en forma de contenido de la imagen.

Erosión para imágenes en escala de grises

Para el caso de las imágenes en escala de grises es posible llevar a cabo la operación de erosión, tomando el valor de menor intensidad que se encuentre dentro de la estructura de referencia, esto se presenta en la ecuación (2.7). El resultado de esta operación genera el mismo efecto que el filtro de erosión para imágenes binarias, reduciendo estructuras de gran tamaño y eliminando las de menor dimensión.

$$E(I) = min_{(s,t)\in H}\{I(i-s,j-t)\} \tag{2.7}$$

Dilatación para imágenes en escala de grises

Al igual que la erosión para imágenes en escala de grises. En la operación de dilatación se toma en cuenta el valor de intensidad de los píxeles contenidos dentro de una estructura de referencia, tomando para esta operación el píxel con mayor nivel de intensidad, como se muestra en la ecuación (2.8).

$$D(I) = max_{(s,t)\in H}\{I(i-s,j-t)\} \tag{2.8}$$

Operación de apertura

Esta operación involucra la implementación del filtro de erosión, y a la imagen resultante se le aplica el filtro de dilatación, tal como se presenta en la ecuación (2.9). Cabe mencionar que esta operación puede ser implementada tanto para imágenes binarias, como para imágenes en escala de grises, empleando para cada tipo de imagen su respectiva estructura de referencia.

$$A(I) = D(E(I)) \tag{2.9}$$

Tonalidad HSV

Como ya se mencionó anteriormente, la forma habitual en que se obtienen las imágenes digitales es mediante el espacio de color RGB, sin embargo, existen otros tipos de tonalidades en las que es posible tratar a las imágenes, como por ejemplo el espacio de color HSV (*Hue Saturation Value*). Este modelo de color consiste en un sistema coordenado cilíndrico (ver figura 2.1) que se puede obtener a través de una imagen RGB por medio de las ecuaciones (2.10), (2.11) y (2.12).

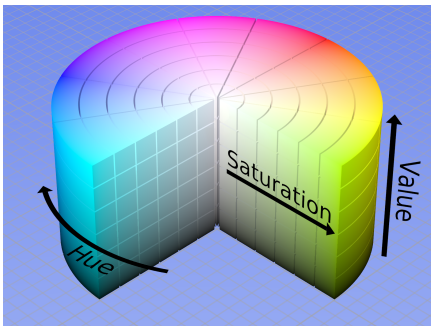


Figura 2.1: Representación del espacio de color HSV.

$$H = arcos \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-G)(G-B))}} \tag{2.10}$$

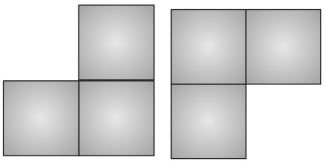
$$S = 1 - 3\frac{min(R,G,B)}{R+G+B} \tag{2.11}$$

$$V = \frac{1}{3}(R+G+B) \tag{2.12}$$

Etiquetado de objetos

El proceso de etiquetado se lleva a cabo en imágenes binarias, éste cuantifica y asigna un valor numérico a cada objeto dentro de una imagen, generando una nueva (E), la cual contendrá los objetos etiquetados. A continuación, se expone el método de etiquetado iterativo: primeramente se recorre la imagen I_b píxel a píxel, y se asigna un valor a cada píxel que tenga intensidad de 1, comenzando desde 1 para el primer píxel que cumpla esta condición, y se va incrementando

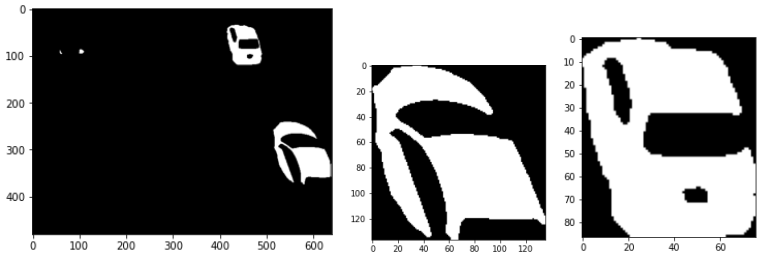
en uno hasta terminar de enumerar todos los píxeles de la imagen. Posteriormente, utilizando una estructura como las presentadas en la figura 2.3, se lleva a cabo un segundo recorrido iterativo por la imagen E, pero empleando la estructura del inciso (a), cuando la imagen se recorre de izquierda a derecha y de arriba hacia abajo, mientras que la estructura del inciso (b), se emplea cuando se hace el recorrido de derecha a izquierda y de abajo hacia arriba.



(a) (b)

Figura 2.2: Estructuras para etiquetado de la imagen de forma iterativa (a) estructura para recorrer imagen de izquierda a derecha y de arriba hacia abajo, (b) estructura para recorrer la imagen de derecha a izquierda y de abajo hacia arriba.

Este segundo recorrido, se lleva a cabo de manera iterativa y asigna un valor al píxel actual, de acuerdo al valor mínimo contenido en la estructura empleada al momento del recorrido. Ésto genera una nueva imagen con los objetos etiquetados. En la figura 2.2 se muestran los objetos (b) y (c), resultado del etiquetado de la imagen (a).



(a) (b) (c)

Figura 2.3: Resultado del etiquetado; (a) imagen segmentada, (b) objeto etiquetado con el numero 1, (c) objeto etiquetado con el numero 5.

En el algoritmo 2.1.1, se presenta el pseudocódigo para llevar a cabo el proceso de etiquetado de objetos en una imagen binaria I_b.

Descriptores

Para que una máquina pueda reconocer objetos en una imagen, éstos deben ser cuantificados. Los descriptores de objetos se encargan de cuantificar los rasgos de un objeto dentro de

Algoritmo 2.1.1: Etiquetado iterativo.

Input: Imagen binaria I_b, fondo en valor 0.

Output: Imagen etiquetada $\rightarrow E$

1 Generar una imagen de indexación $E = 0$ del tamaño de la imagen I_b;
2 Recorrer I_b, y asignar a cada píxel un valor numérico, desde 1 hasta n, el resultado asignarlo a E;
3 Recorrer E, de izquierda a derecha, de arriba hacia abajo y viceversa, hasta que E no cambie;
4 **while** *cambio=1* **do**
5 Recorrer E de izquierda a derecha y de arriba hacia abajo, y asignar el valor mínimo contenido por los píxeles de la estructura al píxel actual $E_{i,j}$;
6 Recorrer E de derecha a izquierda y de abajo hacia arriba, asignar el valor mínimo contenido por los píxeles de la estructura al píxel actual $E_{i,j}$;
7 Determinar si el valor de los píxeles en la imagen E ha cambiado, si no ha cambiado asignar a la variable $cambio = 0$;
8 **end**

una imagen. Las características de un objeto son usualmente almacenadas en un vector, el cual contendrá las características del mismo, y podrá ser utilizado para fines de su clasificación o reconocimiento. Las características deben ser robustas e influenciables a cambios irrelevantes en la imagen, tales como la rotación, desplazamiento y escalado [7]. A continuación, se presentan algunas características empleadas para este trabajo, las cuales son: la firma ($G[n]$), la matriz de co-ocurrencia (GLCM) y algunas características geométricas.

La firma $(G[n])$

Una de las características de las redes PCNN es que pueden generar la firma de un objeto, y debido a que esta es robusta a variaciones de escalado y rotación, es posible utilizarla para el reconocimiento de objetos tal como en [1] y [3]. La firma de un objeto representada por $G[n]$ es producto de la sumatoria de todas las neuronas activadas en una iteración $[n]$ durante el funcionamiento de la red PCNN y se define matemáticamente mediante la expresión de la ecuación (2.13).

$$G[n] = \sum_{i,j} Y_{[i,j]}[n] \tag{2.13}$$

La matriz de co-ocurrencia (GLCM)

La matriz de co-ocurrencia (GLCM) propuesta por Haralick en [13], proporciona información de la relación espacial que existe entre los valores de intensidad de una imagen (para imágenes de 8

bits se tendrá un total de 255 niveles de intensidad), separadas por una distancia d y a lo largo de un ángulo de dirección θ, y se define por la ecuación (2.14).

$$c(i,j) = \left|[(x_1,y_1),(x_2,y_2)], x_2 - x_1 = dcos\theta, y_2 - y_1 = dsen\theta, f(x_1,y_1) = i, f(x_2,y_2) = j\right| \quad (2.14)$$

donde $f(x_1,y_1)$ y $f(x_2,y_2)$, son dos píxeles de la imagen que se este procesando, y $||.||$, denota el número de pares de píxeles que satisfacen la condición. Es decir, que la matriz de co-ocurrencia contendrá la cantidad de veces que se repiten los pares de píxeles que son evaluados con las condiciones de d y θ, previamente establecidas. A partir de la GLCM se pueden obtener rasgos de textura, tales como, contraste, disimilitud, homogeneidad, energía y correlación. Para obtener estas características se debe normalizar la matriz de co-ocurrencia haciendo uso de la ecuación (2.15).

$$P_{i,j} = \frac{c(i,j)}{\sum_{i,j=0}^{N-1} c(i.j)} \quad (2.15)$$

donde $P_{i,j}$, es el valor de probabilidad registrado para la celda i,j dentro de la GLCM; N es el número de filas o columnas. Con la GLCM normalizada se pueden obtener las características indicadas por la ecuaciones (2.16, 2.17, 2.18, 2.19, 2.20, 2.21).

$$constraste = \sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2 \quad (2.16)$$

$$Disimilitud = \sum_{i,j=0}^{N-1} P_{i,j}|i-j| \quad (2.17)$$

$$Homogeneidad = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1-(i-j)^2} \quad (2.18)$$

$$Energía = \sqrt{\sum_{i,j=0}^{N-1} P_{i,j}^2} \quad (2.19)$$

$$Segundo\ momento\ angular\ (ASM) = (\text{Energía})^2 \quad (2.20)$$

$$Correlación = \sum_{i,j=0}^{N-1} P_{i,j}\left\{\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right\} \quad (2.21)$$

donde μ_i, μ_j, son la media de los píxeles de referencia ($P_{i,j}$); σ_i^2, σ_j^2 es la varianza de la GLCM normalizada.

Circularidad

La circularidad de un objeto indica que tan parecido es a un circulo, y los valores en que se cuantifica están comprendidos entre 0 y 1, siendo 1 el valor para un objeto circular mientras que el 0 será para el caso contrario. Para obtener esta característica se requiere de extraer el área y perímetro del objeto, en las ecuaciones 2.22, 2.23 y 2.24 se describe la obtención de la circularidad de un objeto.

$$Perímetro = \sum_{i=1}^{M} longitud(c) \tag{2.22}$$

$$Área = \sum_{i=1}^{f} \sum_{j=1}^{c} I_{b_{i,j}} \tag{2.23}$$

$$Circularidad = 4\pi \frac{área}{perímetro^2} \tag{2.24}$$

2.2. Las redes neuronales pulso-acopladas (PCNN)

Perteneciente a la tercera generación de redes neuronales artificiales debido a que no requieren de un entrenamiento previo para su operación, las Redes Neuronales Pulso-Acopladas o PCNN del inglés *Pulse-Coupled Neural Networks*, también llamadas Redes Neuronales Pulsantes, están basadas en el modelo cortical propuesto por Eckhorn, el cual a su vez está inspirado en la sincronización de pulsos dentro de la corteza visual de los mamíferos. Las PCNN han sido aplicadas ampliamente en el procesamiento de imágenes y reconocimiento de patrones [19].

Si el estímulo en la PCNN es una imagen, entonces la red estará compuesta por tantas neuronas como píxeles se tengan en dicha imagen, y la salida de la red representa características fundamentales de la imagen original, tales como bordes y textura, como se muestra en la figura 2.4. Una neurona pulso-acoplada recibe información de sus neuronas vecinas dentro de la red a través las sinapsis y son disparadas de forma síncrona en ciertas regiones, es por esto que la red PCNN puede ser aplicada en la segmentación, suavizado y codificado de imágenes. Otra importante característica de esta red, es que los pulsos de imágenes pueden ser caracterizados en una firma única para la recuperación e identificación de imágenes. Algunas ventajas a destacar sobre otros métodos, es qué por la naturaleza de esta red es posible realizar el procesamiento en paralelo, mientras qué en los algoritmos tradicionales se suele procesar la imagen píxel a píxel o por regiones, donde se hace notorio el tiempo de procesado a medida en que el tamaño de la imagen aumenta, otro aspecto importante es que la imagen conserva propiedades de escalado y rotación [20].

Debido al potencial de esta red se han desarrollado variantes con base en el modelo original, ya sea simplificando su estructura o mejorando su desempeño. Uno de los modelos que simplifican la PCNN es el Modelo de Intersección Cortical (ICM) del inglés *Intersection Cortical Model* [11], que busca reducir el costo computacional mediante la eliminación de parámetros, del modelo PCNN original [28].

Uno de los puntos clave al momento de procesar imágenes con el modelo PCNN es la selección correcta de sus parámetros, ya que al no presentar una fase de aprendizaje, demanda un ajuste de sus parámetros, siendo para el modelo PCNN original un total de 7, mientras que para el ICM son un total de 3. Para solucionar este problema se han propuesto métodos de sintonización basados en la entropía de los pulsos de imágenes producidas por la red [34], en algoritmos genéticos, que buscan los valores adecuados de los parámetros dado una imagen de referencia previamente segmentada de forma manual [14], [10], o algoritmos iterativos [18].

A continuación, se presentan los modelos de Redes Neuronales Pulso-Acopladas, PCNN e ICM, y su algoritmo para implementación.

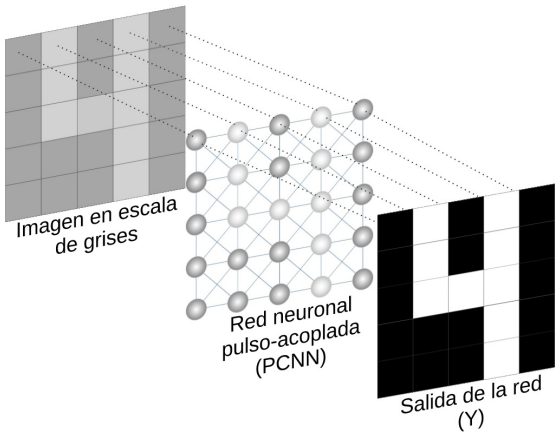


Figura 2.4: Representación de red PCNN.

El modelo PCNN

Una neurona del modelo PCNN original está compuesta por dos canales receptivos denominados *Feeding* (F_{ij}) y *Linking* (L_{ij}), cuyas respuestas son combinadas para generar el potencial interno denominado por U_{ij}, qué posteriormente es comparado con un umbral dinámico θ_{ij}. Cuando U_{ij} sobrepasa el valor de θ_{ij} se produce un disparo, generando una salida Y_{ij}, binaria con valor 1, y de lo contrario, si el potencial interno no rebasa el nivel de umbral, Y_{ij} es 0. En la figura 2.5 se presenta una neurona de la red PCNN, la cual recibe por el canal receptivo F_{ij}, el valor de intensidad de un píxel S_{ij}, correspondiente a la imagen estímulo en escala de grises S, y también recibe a través de las sinapsis definidas por Y_{kl}, la información de sus neuronas vecinas, las cuales se multiplican por su respectiva matriz de pesos y un coeficiente de amplificación, qué para el caso

del canal receptivo L_{ij}, son W_{kl} y V_L respectivamente (ecuación 2.26), y para el canal F_{ij} son M_{kl} y V_F (ecuación 2.25), posteriormente se genera el potencial interno de la neurona U_{ij}, mediante el producto de la salida del canal L_{ij} multiplicado por el factor β, que determina la fuerza de enlazamiento, más la unidad, y la salida del compartimiento F_{ij} (ecuación 2.27), el resultado de ésto se compara con el umbral dinámico θ_{ij} (ecuación 2.29), y si es mayor, la neurona se dispara con valor 1, de lo contrario permanece desactivada (con valor 0), por último, se ajusta el valor de θ_{ij} de acuerdo al coeficiente de amplificación V_θ y la constante de decaimiento $e^{-\alpha T}$ (ecuación 2.28).

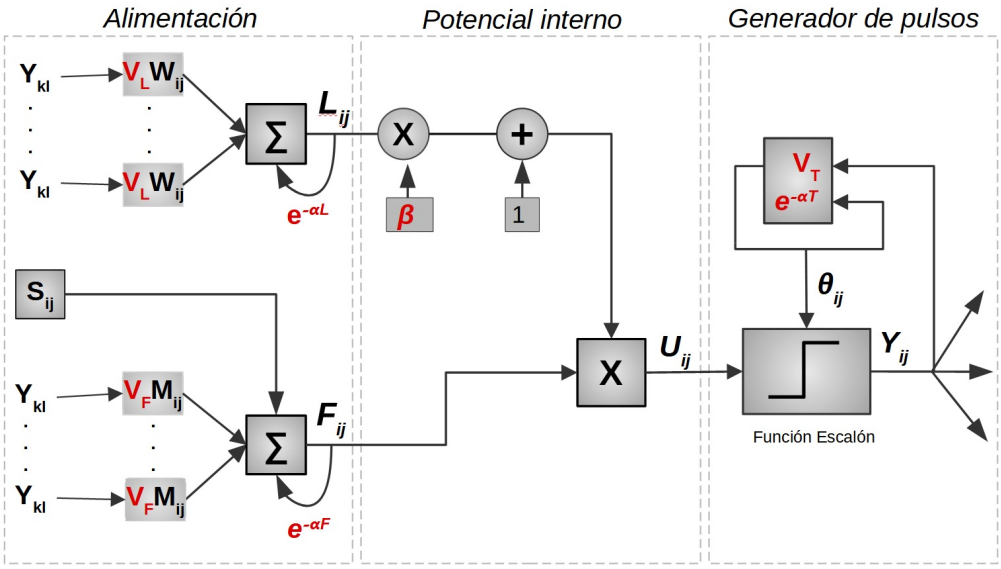


Figura 2.5: Modelo de una neurona tipo PCNN.

$$F_{ij}[n] = e^{\alpha_F} F_{ij}[n-1] + V_F \sum_{kl} M_{ijkl} Y_{kl}[n-1] + S_{ij} \tag{2.25}$$

$$L_{ij}[n] = e^{\alpha_L} L_{ij}[n-1] + V_L \sum_{kl} M_{ijkl} Y_{kl}[n-1] \tag{2.26}$$

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n]) \tag{2.27}$$

$$\theta_{ij}[n] = \theta_{ij}[n-1] e^{-\alpha_T} + V_T Y_{ij}[n-1] \tag{2.28}$$

$$Y[n] = \begin{cases} 1 & \text{si} \quad U_{ij}[n] > \theta_{ij}[n] \\ 0 & \text{en caso contrario} \end{cases} \tag{2.29}$$

$$M = W = \begin{pmatrix} 0{,}25 & 0{,}5 & 0{,}25 \\ 0{,}5 & 0 & 0{,}5 \\ 0{,}25 & 0{,}5 & 0{,}25 \end{pmatrix} \tag{2.30}$$

Algoritmo de la red PCNN

A continuación, en el algoritmo 2.2.1 se presenta el proceso para la implementación del modelo PCNN original. Donde se tiene como entrada una imagen en escala de grises, y como salida los pulsos de imágenes generados por la red.

Algoritmo 2.2.1: Red PCNN.

1 Adquirir imagen en escala de grises $\rightarrow I$;
2 Normalizar la imagen I entre $0-1 \rightarrow S$;
3 Establecer el valor de los parámetros $e^{-\alpha_F}$, V_F, e^{α_L}, V_L, β, $e^{-\alpha_T}$ y V_T;
4 Indicar el número de iteraciones (pulsos) para la red;
5 **for** *iteración* **do**
6 Calcular el canal receptivo *Feeding* utilizando la ecuación 2.25;
7 Calcular el canal receptivo *Linking* utilizando la ecuación 2.26;
8 Calcular el potencial interno U utilizando la ecuación 2.27;
9 Calcular el umbral dinámico θ utilizando la ecuacion 2.28;
10 Obtener la salida de la red Y haciendo uso de la ecuación 2.29;
11 Almacenar el pulso obtenido por cada iteración $Y_i \rightarrow Y$;
12 **end**
13 **return** *Pulsos generados por la red* $\rightarrow Y$

Modelo de intersección cortical (ICM)

Este modelo de red pulso-acoplada es un caso especial del modelo PCNN original, donde el valor de la fuerza de enlace (β) para el canal receptivo *Linking*, es establecido en valor 0, por lo tanto el potencial de activación U es dado únicamente por el canal *Feeding*. En consecuencia a ésto, la cantidad de parámetros se ve reducida a tres, los cuales son: el factor de decaimiento del canal receptivo *Feeding* (***f***), el factor de decaimiento del umbral dinámico (***g***) y ***h***, que determina el incremento del umbral cuando una neurona es disparada. La reducción de los parámetros, hace que el modelo ICM tenga un costo computacional menor en comparación con el modelo PCNN original.

Una neurona del modelo de ICM se presenta en la figura 2.6. Este modelo elimina el canal receptivo *Linking* por lo tanto el potencial interno de la neurona viene dado unicamente por el canal *Feeding*, en donde se recibe el estímulo $S_{i,j}$, que corresponde a un valor de intensidad (píxel) de la imagen estímulo S, y la información de las neuronas vecinas Y_{kl}, que son ponderadas con un peso respectivo dado por la matriz M (ecuación 2.34). El funcionamiento de la red ICM viene dada por las ecuaciones 2.31 a 2.34.

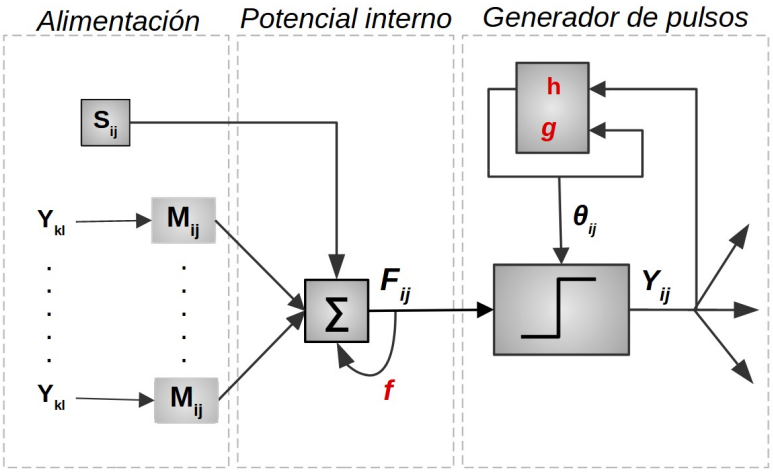


Figura 2.6: Modelo de neurona tipo ICM.

$$F_{ij}[n] = fF_{ij}[n-1] + \sum_{kl} W_{ijkl}Y_{kl}[n-1] + S_{ij} \tag{2.31}$$

$$\theta_{ij}[n] = g\theta_{ij}[n-1] + hY_{ij}[n-1] \tag{2.32}$$

$$Y_{ij}[n] = \begin{cases} 1 & \text{si} \quad F_{ij}[n] > \theta_{ij}[n] \\ 0 & \text{en caso contrario} \end{cases} \tag{2.33}$$

$$W = \begin{pmatrix} 0{,}25 & 0{,}5 & 0{,}25 \\ 0{,}5 & 0 & 0{,}5 \\ 0{,}25 & 0{,}5 & 0{,}25 \end{pmatrix} \tag{2.34}$$

Algoritmo de la red ICM

En el algoritmo 2.2.2, se presenta el procedimiento para la implementación del modelo ICM, en donde se recibe una imagen en escala de grises, y como salida se tienen los generados por la red.

Algoritmo 2.2.2: Red ICM.

Adquirir imagen en escala de grises $\rightarrow I$;
Normalizar la imagen I entre $0-1 \rightarrow S$;
Establecer el valor de los parámetros f, g y h;
Indicar el número de iteraciones (pulsos) para la red;
for *iteración* **do**
    Calcular el canal *Feeding* utilizando la ecuación 2.31;
    Calcular el umbral dinámico θ utilizando la ecuación 2.32;
    Obtener la salida de la red Y haciendo uso de la ecuación 2.33;
    Almacenar el pulso obtenido por cada iteración $Y[i] \rightarrow Y$;
end
return *Pulsos generados por la red* $\rightarrow Y$

2.3. Aprendizaje máquina

El aprendizaje máquina o *Machine Learning*, pertenece al conjunto de áreas que conforman la inteligencia artificial y se puede definir como la ciencia de programar computadoras de tal forma que puedan aprender de los datos que se le proporcionen.

También se puede referir como un programa flexible que va descubriendo como resolver el problema en base a ejemplos. Algunas de sus principales aplicaciones son:

- Clasificación, permite la identificación de objetos considerando un número limitado de posibilidades.
- Regresión, al igual que la clasificación lleva a cabo el proceso de identificación de objetos, pero en este caso se considera un número de posibilidades ilimitado.

Una parte importante del aprendizaje máquina, es el proceso de aprendizaje mismo, el cual se puede llevar a cabo bajo las dos principales formas siguientes:

- Aprendizaje supervisado, va alimentando al algoritmo con ejemplos y adjuntado a cada ejemplo se asigna una etiqueta que indica la salida deseada a obtener.
- Aprendizaje no supervisado, no requiere de etiquetas que representen las salidas deseadas, en cambio el algoritmo aprende automáticamente de las muestras recibidas.

A continuación, se presentan algunos de los algoritmos utilizados en aprendizaje automático, los cuales son: el algoritmo k vecinos más cercanos (*KNN*) y el Perceptrón Multicapa (*MLP*), cada uno con su respectivo método de aprendizaje.

K vecinos más cercanos (kNN)

El clasificador k-vecinos más cercanos o kNN del inglés *k-nearest neighbors*, es un clasificador basado en aprendizaje supervisado, el cual consiste en asignarle el valor a un punto objetivo (o de consulta) dependiendo de sus k vecinos más cercanos, a los cuales se les asignó una etiqueta durante la etapa de aprendizaje. Su forma de operar se basa en identificar los k vecinos más cercanos al punto de consulta, y posteriormente seleccionar la etiqueta que más se repita del conjunto de vecinos.

Para identificar la cercanía desde un punto de consulta, hacia los demás puntos correspondiente al conjunto de muestras de aprendizaje, se suele utilizar el cálculo de la distancia entre éstos, posteriormente el algoritmo elige los k vecinos más cercanos (en distancia), y de este conjunto de vecinos se elige la clase que más se repite o que sea dominante. A ésto se le suele llamar una clasificación por votación, ya que la clase ganadora será aquella que cuente con mayor número de votos por los vecinos [21], [26].

El clasificador kNN lo podemos dividir en dos etapas, la primera será la fase de aprendizaje, que consiste en almacenar la información (muestras), y la segunda fase es la etapa de clasificación, donde se recibe el punto de consulta ó muestra a clasificar, y se compara con las demás muestras de entrenamiento (en distancia), asignando la clase que más se repita entre sus k vecinos. Para identificar la cercanía de un punto de consulta con las muestras de aprendizaje, se suele emplear la distancia Euclidiana definida en la ecuación (2.35).

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.35}$$

Donde d es la distancia cálculada, x_i es el n vector de características del conjunto de aprendizaje y y es el nuevo vector de características a clasificar. A continuación, en el algoritmo 2.3.1, se describen los pasos para la implementación del clasificador *kNN*.

Algoritmo 2.3.1: Algoritmo k vecinos más cercanos.

1 ***Fase de entrenamiento***
2 Agregar datos de entrenamiento;
3 Asociar una clase a cada muestra del vector de entrenamiento $(x, f(x))$;
4 ***Fase de clasificación***
5 **for** *cada nueva imagen a clasificar* x_q **do**
6 | Calcular la distancia Euclidiana entre x_q y $x_i \rightarrow dist = \sqrt{\sum_{i=1}^{n}(x_i - x_q)^2}$;
7 **end**
8 Determinar las distancias más cercanas entre los atributos x_q y x_i dadas por $dist$, linea 6 ;
9 Seleccionar las clases de las distancias cercanas dado el valor de k ;
10 Seleccionar la clase que más se repite dado el conjunto generado en línea 9 ;
11 **return** *Clase*

Perceptrón multicapa (MLP)

La red perceptrón multicapa o *Multi-Layer Perceptron MLP*, está compuesta por un arreglo de neuronas de tipo perceptrón ordenados por capas [12]. Una MLP presenta al menos tres capas de neuronas: capa de entrada, oculta y de salida. Así mismo, esta red tiene al menos dos neuronas en cada capa oculta y una neurona en las capas de entrada y salida. El perceptrón multicapa, tiene un amplio rango de aplicación en multitud de problemas de diferentes áreas del conocimiento, y es considerado una de las arquitecturas de redes neuronales más versátiles [8]. Algunas de sus principales aplicaciones son: aproximador de funciones universal, reconocimiento de patrones, control y procesos de identificación. En la figura 2.7 se muestra la representación de una red MLP, la cual cuenta con entradas $(x_1, x_2, ..., x_n)$ en la capa de entrada (C_{en}); una capa oculta (C_{oc}), con n_1 neuronas, así como una capa de salida (C_{sal}) con y_m neuronas.

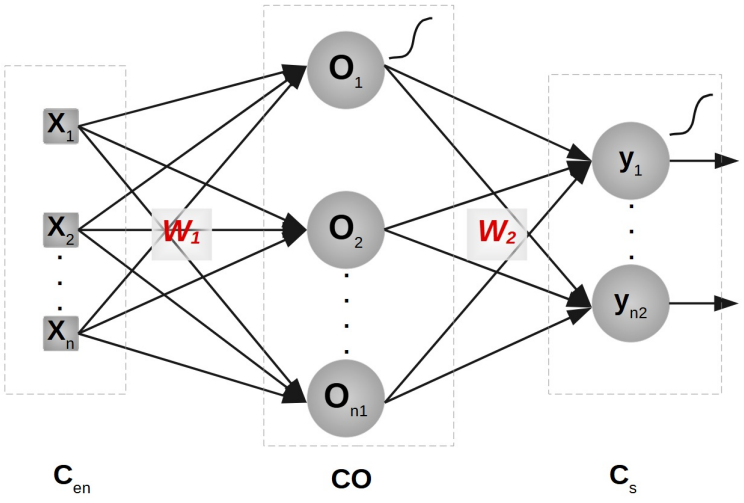


Figura 2.7: Arquitectura de la red perceptrón multicapa.

Para enlazar las neuronas de C_{en} con las de C_{oc}, se emplea la matriz w_1 que representa la sinápsis; por otro lado, la matriz w_2 representa las sinápsis que transporta la información de cada neurona en C_{oc} hacia las neuronas de C_{sal}. A cada componente de una matriz w, se le denomina peso sináptico, mismo que se multiplica por la señal de entrada de la sinápsis, la cual puede provenir de una neurona en C_{en} o de C_{oc}, según sea el caso.

Proceso de aprendizaje

El proceso de aprendizaje para la red MLP se denomina *backpropagation* [27], el cual se puede dividir en tres partes: la primer parte es la propagación hacia adelante (*forward propagation*), que consiste en generar la salida de la red a través de la activación de sus neuronas; la segunda es la retropropagación (*backpropagation*), en donde se lleva a cabo el ajuste de los pesos sinápticos de las matrices w, en base al error cometido por la red; y por último, se lleva a cabo el proceso de evaluación, que trata de generar el error global de la red, y si dicho error sobrepasa un nivel de umbral de precisión definido por ε, se detiene el proceso de aprendizaje. A continuación se describen cada una de estas etapas que conforman el algoritmo *backpropagation*.

Propagación hacia adelante

La propagación hacia adelante consiste en producir las salidas de cada neurona mediante la ecuación (2.36), y posteriormente acotar dicha salida mediante una función de activación, como las que se presentan en la ecuación (2.37). Este proceso comienza con las neuronas de la primer capa oculta, hasta llegar a las neuronas de la capa de salida.

$$u = \sum_{i=1}^{n} (w_n * x_n) - \theta \tag{2.36}$$

$$G(u) = tanh(u) \text{ ó bien } G(u) = sigmoide(u) \tag{2.37}$$

donde u representa el potencial de activación de una neurona; n indica el número de entradas (sinápsis de la neurona), que para el caso de las neuronas de C_{oc}, representa la cantidad de características (x_n) dado por un vector de entrada x, mientras que para las neuronas de C_{sal}, corresponderá al número de neuronas en la capa C_{oc}; y $G(u)$ representa la salida o activación de dicha neurona, que está dada por una función de activación, la cual puede ser sigmoidal o tangente inversa, ésto se hace para acotar el valor de salida entre 0 y 1 ó -1 y 1 respectivamente.

Una vez generada la salida de la red se concluye el proceso de propagación hacia adelante, y se da paso a la segunda etapa, que consiste en la retropropagación del error, para corregir el valor de los pesos sinápticos w.

Retropropagación

Esta etapa consiste en modificar el valor de los pesos sinápticos, comenzando con los que se encargan de enlazar las neuronas de la capa C_{oc} con las neuronas de la capa C_{sal} definido por la matriz w_2, y posteriormente se continúa con los de la matriz w_1. Inicialmente se obtiene el error de la red dada una muestra x, mediante la diferencia entre la salida deseada d y la producida por la red y (ecuación 2.38).

$$e = d - y \tag{2.38}$$

Obteniendo la medida del error (e), se procede a calcular $\delta_{C_{sal}}$, que corresponde al error producido por la red dado un cambio generado en el potencial u de una neurona en la capa C_{sal}, con un conjunto de pesos sinápticos w_2 (ecuación 2.39).

$$\delta_{C_{sal}} = \frac{\partial y}{\partial u} * e_i \tag{2.39}$$

En seguida, se lleva a cabo el ajuste de los pesos sinápticos w_2, tal como se índica en la ecuación (2.40).

$$w_2 = w_2 + (\lambda * \delta_{C_{sal}} * y_i) \tag{2.40}$$

donde λ, es el factor de aprendizaje, el cual puede tener un valor entre 0 y 1.

Por ultimo, realiza el ajuste del umbral $\theta_{C_{sal}}$ correspondiente a las neuronas de salida y, el cual se calcula mediante la ecuación (2.41).

$$\theta_{C_{sal}} = \theta_{C_{sal}} + (\lambda * \delta_{C_{sal}}) \tag{2.41}$$

Ahora se continua con el ajuste de los pesos sinápticos de la matriz w_1. Para ésto se debe obtener $\delta_{C_{oc}}$, que corresponde al cambio producido en el potencial u de las neuronas de la capa C_{oc} ante una entrada x con pesos sinápticos w_1, y para transportar la medida del error cometido por la red (e), se hace uso del valor $\delta_{C_{sal}}$, ecuación (2.42).

$$\delta_{C_{oc}} = \frac{\partial G(u)}{\partial u} * w_2 * \delta_{C_{sal}} \tag{2.42}$$

El ajuste de w_1 esta definido por la ecuación (2.43).

$$w^{(1)} = w^{(1)} + (\lambda * \delta_{C_{oc}} * x) \tag{2.43}$$

Por ultimo se ajusta el umbral ($\theta_{C_{oc}}$), utilizando la ecuación (2.44).

$$\theta_{C_{oc}} = \theta_{C_{oc}} + (\lambda * \delta_{C_{oc}}) \tag{2.44}$$

Una vez terminado el ajuste de los pesos w y θ respectivos a las capas C_{sal} y C_{oc}, se vuelve a llevar a cabo el proceso de propagación hacia adelante pero ahora considerando los nuevos valores ajustados, con lo cual se espera obtener una salida y, que se aproxima de mejor manera que en un tiempo anterior. En consecuencia se producirá un error diferente al obtenido previamente, el cual se utilizará para generar el error global de la red E, a continuación se presenta este proceso.

Evaluación

Comienza cuantificando el error cuadrático medio ($e(k)$) cometido por la red dado una entrada x, correspondiente a una muestra i del conjunto de muestras X (ecuación 2.45).

$$e(k)_i = \frac{1}{2}(d_i - y_i)^2 \tag{2.45}$$

El resultado de $e(k)$ se almacena para cada muestra i dentro del conjunto X, y al finalizar se hace el promedio de todos los errores cometidos para generar la medida del error global de la red E (ecuación 2.46).

$$E = \frac{1}{p}\sum_{k=1}^{p} e(k) \tag{2.46}$$

donde p corresponde al total muestras x dentro del conjunto X. Por último, se compara el error global E con el valor de error permitido (ε), y si E es menor que ε, entonces se detiene el proceso de aprendizaje, concluyendo que la red ha sido entrenada. Ademas, se suele utilizar el parámetro $epocas$, para detener el proceso de aprendizaje, dado un número de iteraciones máximo, en caso de que la red no logre alcanzar la precisión ε.

2.4. La tarjeta Raspberry Pi 4

Las tarjetas *Raspberry Pi* son pequeñas computadoras diseñadas por la *fundación Raspberry Pi*, para la enseñanza y aprendizaje de la computación digital; actualmente en el mercado se pueden encontrar diferentes modelos de esta mini-computadoras, siendo *Raspberry Pi 4* el modelo más reciente, el cual se puede conseguir en diferentes configuraciones de memoria RAM (2GB, 4GB, 8GB), incorpora un procesador Quad Core de 64 bits ARM-Cortex A72 que corre a una velocidad de 1.5 GHz, cuenta con comunicación inalámbrica Bluetooth 5.0 BLE y Wi-Fi, algunos de los principales componentes de este modelo se presentan en la figura 2.8, el cual corresponde a un modelo de 4GB de RAM.

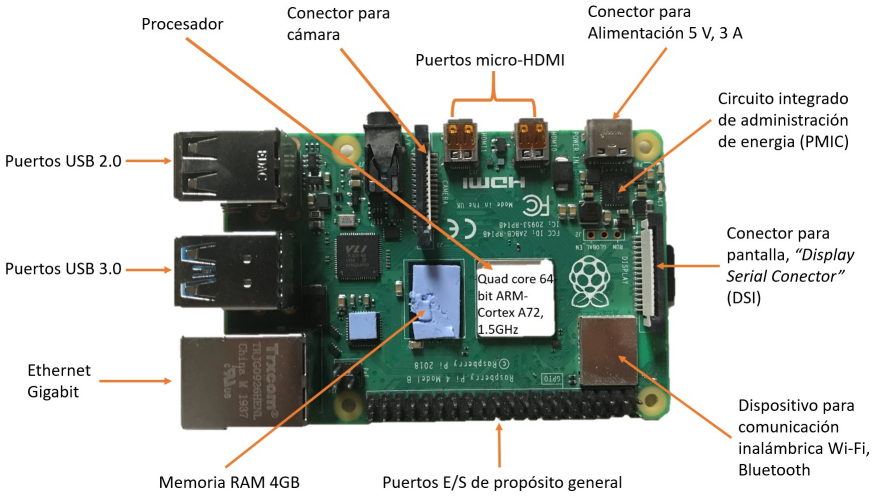
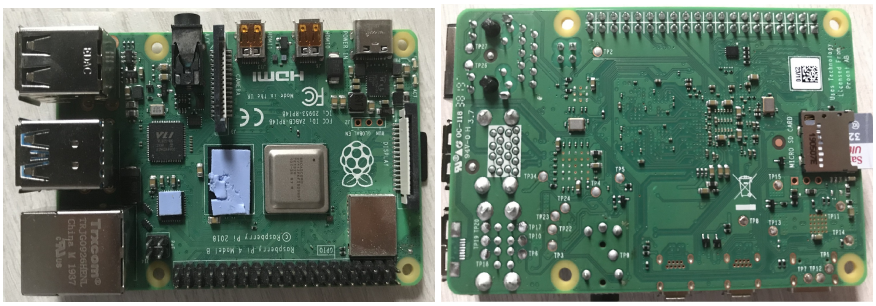


Figura 2.8: Principales componentes de la tarjeta *Raspberry Pi 4*.

Presenta software basado en Linux y un set de instrucciones ARMv8. Los sistemas operativos que pueden ser instalados en esta tarjeta son diversos, algunos de éstos son *Raspberry Pi OS*, que es el sistema operativo oficial, además, se cuenta con *Ubuntu*, *Manjaro ARM Linux*, *RISC OS Pi*, entre otros. Se puede acceder al gestor de descargas en la página `https://www.raspberrypi.org/software`. Cabe mencionar que, a pesar de que este modelo presenta un procesador de 64 bits, la mayoría de los sistemas operativos que ofrece la *organización Raspberry Pi* son para arquitectura de 32 bits, tal es el caso del sistema operativo *Raspberry Pi OS*, el cual se utilizó para llevar a cabo las pruebas con esta tarjeta durante la realización de este trabajo, y que se instaló instalado sobre la tarjeta *Raspberry Pi 4* de la figura 2.9.



(a) (b)

Figura 2.9: Tarjeta *Raspberry Pi 4* que se utilizó para este trabajo.

Capítulo 3

Diseño e implementación del algoritmo

En este capítulo se presenta el trabajo realizado de acuerdo a cada objetivo específico planteado. Se comienza con el primer objetivo que establece la tarea de recolectar las imágenes de un estacionamiento; posteriormente el proceso de segmentación; en seguida se presentan los métodos de extracción de características; se continúa con el método para clasificación basado en redes neuronales; y por último se presenta el desarrollo de la interfaz de usuario.

Los algoritmos que se presentan se implementaron en el lenguaje de programación *Python* versión $3{,}7$; de igual forma, se utilizaron las bibliotecas de *Numpy* para computo matemático, *Matplotlib* para mostrar gráficos e imágenes, *Pillow* para procesamiento de imágenes, y *os* para leer varios archivos de una ruta específica. Las imágenes procesadas están en formato jpg, redimensionadas a tamaño VGA.

3.1. Imágenes adquiridas de un estacionamiento

Como parte del primer objetivo específico de este trabajo, se llevó a cabo la recolección de imágenes del estacionamiento de la Facultad de Ciencias de la Electrónica. La topología del estacionamiento se representa en la figura 3.1, donde se obtuvieron cuatro tomas distintas, las cuales corresponden a la parte derecha, centro, izquierda y lateral, del estacionamiento (ver figura 3.2).

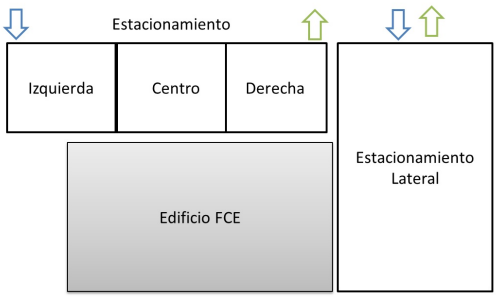


Figura 3.1: Regiones de captura de imágenes de un estacionamiento.

En total se cuenta con un conjunto de 408 imágenes del estacionamiento, con 69 imágenes

(a) (b) (c) (d)

Figura 3.2: Diferentes vistas del estacionamiento. (a) parte derecha; (b) parte centro; (c) parte izquierda; (d) parte lateral.

de la parte lateral, 152 de la parte centro, 81 de la parte izquierda y 106 de la parte derecha. Las cuales pertenecen a una base de imágenes que se tenía en la facultad, ya que debido a la pandemia COVID-19 no fue posible realizar la captura de las mismas.

3.2. Segmentación de las imágenes

Una vez recabadas las imágenes del estacionamiento, como parte del segundo objetivo específico de este trabajo, se llevó a cabo el proceso de segmentación. Éste consiste en aislar los objetos de interés del fondo de la imagen.

En este trabajo los objetos de interés resultan ser los automóviles, es decir que en el proceso de segmentación se buscará obtener una imagen binaria donde los valores de intensidad en 1 correspondan a los objetos de interés, y los valores en 0 al fondo de la imagen (piso del estacionamiento). Este proceso resulta de mayor importancia debido a qué los resultados de detección de espacios disponibles dependerán de una buena segmentación de las imágenes.

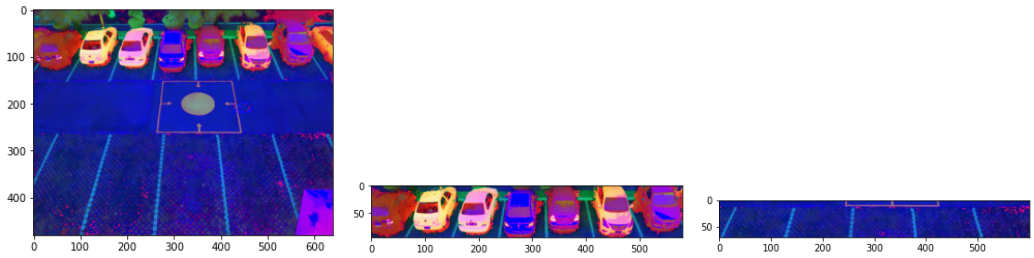
Se llevaron a cabo pruebas de segmentación con el conjunto de imágenes del estacionamiento, haciendo uso de los espacios de color RGB, HSV y escala de grises. De acuerdo con estas pruebas, se observó que las imágenes en espacio de color HSV proporcionan mejor información de los objetos de interés; debido a esto se diseñaron tres métodos de segmentación basados en el uso de las redes neuronales pulso-acopladas (PCNN) en su forma simplificada, el modelo de intersección cortical (ICM), los cuales son: método de segmentación por zonas, método de segmentación basada en suma de pulsos de máxima entropía y método basado en pulsos máxima entropía. Además, se diseñó un cuarto método, el cual es una alternativa a los tres métodos previos, donde a diferencia de éstos, se lleva a cabo la ubicación de regiones de interés (ROI) en lugar de emplear la segmentación.

A continuación, son expuestos los métodos propuestos de segmentación y el método alternativo que consiste en la ubicación de ROI; los cuales se han presentado como propuestas de trabajo en los congresos: CIITEC 2020, CIRC 2021, SENIE 2021 y CONACIC 2021.

3.2.1. Método de segmentación por zonas

De acuerdo a una serie de pruebas de segmentación con el conjunto de imágenes de estacionamiento empleando los algoritmos 2.2.1 y 2.2.2, se observó que algunos objetos que no son de interés, tales como los árboles y señalamientos en el estacionamiento, son activados por la red al mismo tiempo o antes que los automóviles, dando como resultado una segmentación no deseada. Para evitar lo anterior, en este método se propuso llevar a cabo dos tareas. La primera consiste en dividir la imagen en dos zonas, donde una zona enfoca la parte superior y otra la inferior de la imagen, qué es donde se encuentran los objetos de interés; y en la segunda se llevó a cabo la limitación de colores en el espacio de color HSV.

El método propuesto, considera el uso de las cuatro tomas diferentes del estacionamiento, es decir, la parte derecha, centro, izquierda y lateral, presentadas en la figura 3.2. Inicialmente se recibe una imagen en el espacio de color RGB, y ésta es convertida al espacio de color HSV empleando las ecuaciones (2.10, 2.11 y 2.12). Una vez hecho lo anterior se da lugar a la localización de las zonas donde se encuentran los lugares de estacionamiento. Esta tarea se lleva a cabo manualmente, indicando las coordenadas de cada una de éstas. En la figura 3.3, se muestra el resultado del proceso de ubicación de zonas en una imagen de estacionamiento; el inciso (a) muestra la imagen original en espacio de color HSV; en el inciso (b) se muestra la zona superior; y el inciso (c) la zona inferior.



(a) (b) (c)

Figura 3.3: Resultados del proceso de ubicación de zonas de segmentación. (a) imagen original en espacio de color HSV; (b) zona superior, (c) zona inferior.

Una vez que se han ubicado las zonas en la imagen, se da lugar al proceso de limitación de colores en el canal de tonalidad (Hue); éste canal forma parte del espacio de color HSV. El ajuste consiste en limitar el rango de colores correspondiente a los árboles y piso del estacionamiento, para así resaltar los objetos de interés en la imagen. En la figura 3.4, se muestra este proceso para una imagen del estacionamiento izquierdo, de la cual se ha tomado la zona superior generada en el proceso anterior, misma que se presenta en el inciso (a) de la figura 3.4. Posteriormente se obtiene el canal de tonalidad de dicha zona (inciso (b) de la figura 3.4); para eliminar el ruido se

aplica el filtro de la mediana (inciso (c) de la figura 3.4); enseguida se lleva a cabo la limitación de colores en el canal Hue, para poder eliminar los niveles de intensidad que corresponden a los árboles y estructuras del estacionamiento, ésto se hace mediante la ubicación de un umbral, el cual se establece de forma manual, donde el valor de éste, se obtiene a través de un análisis previo del histograma; el resultado de este proceso se presenta en el inciso (d) de la figura 3.4; como paso siguiente se lleva a cabo la operación de dilatación para rellenar pequeños huecos en las estructuras de los autos (inciso (e) de la figura 3.4), y la imagen resultante es normalizada a valores entre 0 y 1; posteriormente la imagen normalizada es introducida a la red ICM. Los parámetros del modelo ICM se ajustaron a los siguientes valores: $\boldsymbol{f} = 0{,}4$; $\boldsymbol{g} = 0{,}5$; $\boldsymbol{h} = 21$, y el número de iteraciones se estableció en 1, es decir que solo se va a generar un pulso, el cual entregara la imagen segmentada (inciso (f) de la figura 3.4).

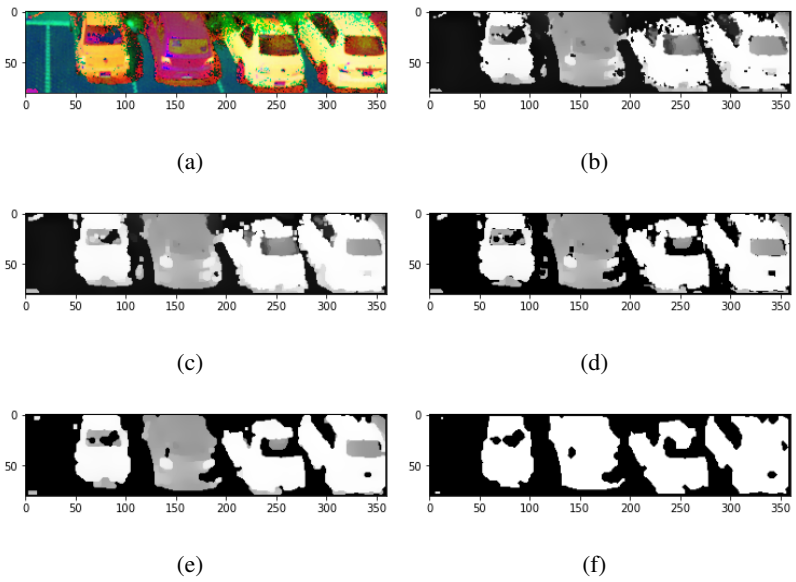


Figura 3.4: Proceso de segmentación por zona en una imagen. (a) zona superior; (b) obtención de canal Hue; (c) aplicación del filtro de la mediana, (d) limitación de tonalidad; (e) aplicación de operación de cierre; (f) zona segmentada por la red ICM.

En el algoritmo 3.2.1, se presenta el método de segmentación por zonas propuesto. El cual recibe una imagen en espacio de color RGB y entrega como resultado las zonas de la imagen segmentadas. Éstas serán empleadas en una etapa siguiente para llevar a cabo la detección de automóviles.

3.2.2. Método de segmentación basado en suma de pulsos de máxima entropía

En la subsección 3.2.1, se presentó el uso de la red ICM para llevar a cabo la segmentación de las imágenes de un estacionamiento, mediante la ubicación de dos zonas de interés, y la limitación

Algoritmo 3.2.1: Algoritmo de segmentación por zonas.

Input: I_{rgb}, $f = 0{,}4$, $g = 0{,}5$, $h = 21$

Result: Zonas segmentadas (S_a, S_b)

1 Convertir I_{rgb} al espacio de color HSV $\rightarrow I_{hsv}$;
2 Ubicar las zonas en la imagen $I_{hsv} \rightarrow I_a, I_b$;
3 Obtener el canal *Hue* de cada sub-imagen I_a, I_b, y aplicar operación de apertura $\rightarrow I_a^{'}, I_b^{'}$;
4 Limitar colores en $I_a^{'}, I_b^{'}$ de acuerdo al umbral $u \rightarrow L_a, L_b$;
5 Normalizar las imágenes $L_a, L_b \rightarrow N_a, N_b$;
6 Aplicar segmentación por ICM a $N_a, N_b \rightarrow S_a, S_b$;
7 Aplicar operaciones de apertura y cierre a S_s, S_b;

de color en el canal Hue. Sin embargo, de acuerdo con los resultados obtenidos, los cuales se presentan en la sección 4.1 del capítulo 4, se detectó que, se puede presentar una mala ubicación de las zonas de interés a causa del movimiento de la cámara de captura, por otro lado, la limitación de color en el canal Hue, puede ser contraproducente, debido a que pueden presentarse automóviles con tonalidad dentro del rango de umbral que lo elimina, lo cual puede contribuir a obtener un bajo desempeño de segmentación. Para reducir los inconvenientes mencionados, se excluyeron los puntos que corresponden a dividir la imagen en dos zonas de interés y la limitación de niveles de intensidad; para evitar la problemática de una mala ubicación de las zonas de interés debido al movimiento de la cámara. Además, se logró sustituir el punto que corresponde a la delimitación de los niveles de intensidad, mediante el ajuste de los parámetros de la red ICM, aprovechando la serie de pulsos de imágenes que arroja esta red, para seleccionar de éstos los que presenten mayor información.

Para este método se eligieron un total de 123 imágenes del conjunto presentado en la sección 3.1. Las cuales corresponden a las imágenes de la parte derecha, centro e izquierda; con al menos 24 imágenes por cada parte del estacionamiento.

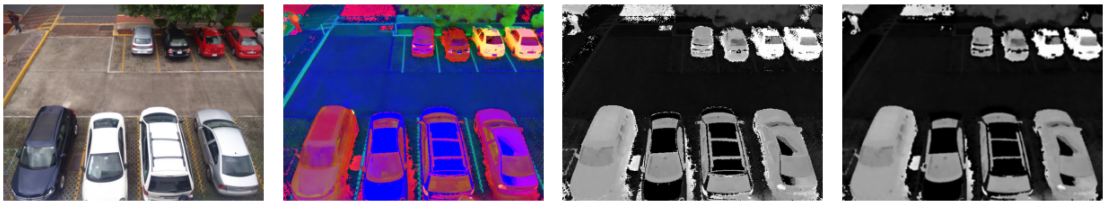
Primeramente se diseñó la etapa de pre-procesamiento, que consiste en la conversión de la imagen original en espacio de color RGB a HSV mediante la relación presentada en las ecuaciones (2.10, 2.11 y 2.12); posteriormente, de la imagen en espacio de color HSV (I_{hsv}), se extrae el canal Hue (I_h). Para eliminar pequeñas partículas en la imagen I_h, se aplica la operación morfológica de apertura para imágenes en escala de grises; esta operación implementa primeramente un filtro de erosión, seguido del filtro de dilatación; en las ecuaciones (2.7) y (2.8) se presenta la relación de estos operadores respectivamente, en donde I representa la imagen del canal Hue; H corresponde al tamaño del filtro utilizado para la operación, dado en filas y columnas; mientras que i, j representa la posición de un píxel en la imagen I; y en la ecuación (2.9) se indica la operación de apertura para una imagen en escala de grises. Para la aplicación de los filtros mencionados se

requiere definir la estructura de referencia, la cual define la influencia que tienen los píxeles vecinos sobre un píxel central. Las estructuras de referencia utilizadas para la aplicación de la operación de apertura, se presentan en las ecuaciones (3.1) y (3.2), para aplicar los filtros de erosión y dilatación respectivamente.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.1}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.2}$$

En la figura 3.5, se presentan los resultados de la etapa de pre-procesamiento; en el inciso (a), se presenta la imagen de entrada; en el inciso (b), se muestra la imagen original en el espacio de color HSV; el inciso (c), presenta el canal de tonalidad (Hue); y la imagen del inciso (d), corresponde al resultado de la operación de apertura.



(a) (b) (c) (d)

Figura 3.5: Etapa de pre-procesamiento. (a) imagen de entrada en RGB; (b) conversión al espacio de color HSV; (c) parte izquierda; (d) parte lateral.

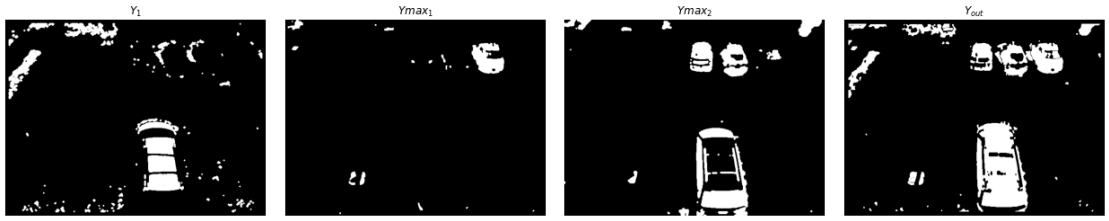
Una vez realizada la etapa de pre-procesamiento, se continua con la fase de segmentación, para ésto se empleó la red ICM. Para la sintonización de los parámetros, se llevó a cabo una serie de pruebas, en donde se dedujeron los siguientes valores: $\boldsymbol{f} = 0{,}19$, $\boldsymbol{g} = 0{,}6$ y $\boldsymbol{h} = 19$. Así mismo la matriz de pesos sinápticos W que se utilizó, se indica en la ecuación (3.3). La cantidad de iteraciones se estableció a nueve, debido a que se observó que a partir de la décima iteración la red comenzaba a repetir el patrón de pulsos de salida $Y[n]$. Para obtener la mejor segmentación, se realizó la integración de tres pulsos, los cuales son: el primer pulso, con el cual se obtiene la información de automóviles en color blanco, y los dos pulsos de mayor entropía, de los cuales se

obtiene la información de las demás tonalidades de color. La entropía se obtiene de la ecuación (3.4), donde p_0 corresponde a la probabilidad de obtener el valor 0 y p_1 para el valor 1. En el algoritmo 3.2.2, se presenta el método de segmentación propuesto.

$$W = \begin{pmatrix} 0{,}1 & 0{,}5 & 0{,}1 \\ 0{,}5 & 0 & 0{,}5 \\ 0{,}1 & 0{,}5 & 0{,}1 \end{pmatrix} \tag{3.3}$$

$$H = -p_0 log_2 p_0 - p_1 log_2 p_1 \tag{3.4}$$

En la figura 3.6, se muestran los resultados del proceso de segmentación propuesto, donde la imagen del inciso (a), corresponde al primer pulso producido por la red, y las imágenes de los incisos (b) y (c), son los pulsos de mayor entropía, y el resultado de segmentación se presenta en la imagen del inciso (d).



(a) (b) (c) (d)

Figura 3.6: Etapa de segmentación. (a) primer pulso generado por la red; (b) pulso de mayor entropía; (c) segundo pulso de mayor entropía; (d) resultado de segmentación.

Algoritmo 3.2.2: Algoritmo de segmentación basado en la suma de pulsos.

Input: I_{rgb}, $f = 0{,}19$, $g = 0{,}6$, $h = 19$

Result: Imagen segmentada S

1 Convertir I_{rgb} a espacio de color HSV $\rightarrow I_{hsv}$;
2 Extraer Hue y aplicar operación de apertura $\rightarrow I_{hue}$;
3 Introducir I_{hue} a red ICM y generar 10 pulsos $\rightarrow S_i$;
4 Obtener el primer pulso y los dos de mayor entropía, y sumarlos $\rightarrow S$;

3.2.3. Método de segmentación basado en pulsos de máxima entropía

El método presentado en la subsección 3.2.2, no toma en cuenta las imágenes de la parte izquierda del estacionamiento, debido a que las estructuras de los autos en la parte superior de

las imágenes son eliminados, a causa de su tamaño, el cual es de menor dimensión que los autos de las imágenes en las partes derecha, centro e izquierda. Debido a ésto se diseñó un algoritmo de segmentación para las imágenes de la parte lateral del estacionamiento.

Este método consiste inicialmente de una etapa de pre-procesamiento, donde se lleva a cabo la conversión de la imagen de entrada al espacio de color HSV, y posteriormente se obtiene el canal de tonalidad (Hue); a la imagen resultante de lo anterior, se le aplica la operación de apertura, usando una estructura de referencia cuadrada de tamaño 3x3 píxeles. El tamaño de la estructura de referencia se eligió, debido a que ésta permite eliminar pequeños elementos en la imagen de manera adecuada, sin afectar las estructuras de los automóviles de menor tamaño.

(a) (b) (c)

Figura 3.7: Etapa de pre-procesamiento. (a) imagen de entrada; (b) conversión al espacio de color HSV; (c) extracción del canal Hue y aplicación de operación de apertura.

Para llevar a cabo el proceso de segmentación con la red ICM, se modificó el proceso para obtener la información de las neuronas vecinas en el canal $Feeding$. Dicha modificación se indica en la ecuación (3.5), donde se eliminó la operación de convolución entre $Y[n-1]$ y W, de manera directa, en cambio, se optó por realizar esta operación por separado, incorporando el parámetro $conv$ al modelo; para que cuando el resultado de la convolución sea mayor a cero, se asigne el valor $conv$ a $Y \otimes W$, como se muestra en la ecuación (3.8). Con esta modificación se obtiene un mayor control del funcionamiento de la red ICM, ya que es posible predecir con mayor facilidad las neuronas que serán activadas en cada iteración. Las ecuaciones (3.6) y (3.7), presentan la obtención del umbral dinámico y la salida de la red respectivamente, mientras que la ecuación (3.9), presenta la matriz de pesos sinápticos empleada.

$$F_{ij}[n] = fF_{ij}[n] + S_{ij} + Y \otimes W \tag{3.5}$$

$$\theta_{ij}[n] = gT_{ij}[n-1] + hY_{ij}[n-1] \tag{3.6}$$

$$Y_{ij}[n] = \begin{cases} 1 & \text{si} \quad F_{ij}[n] > \theta_{ij}[n] \\ 0 & \text{en caso contrario} \end{cases} \tag{3.7}$$

$$Y \otimes W = \begin{cases} conv, & \sum_{kl} W_{ijkl} Y_{kl}[n-1] > 0 \\ 0, & \text{en caso contrario} \end{cases} \tag{3.8}$$

$$W = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.9}$$

Los parámetros de la red ICM se ajustaron a los siguientes valores: $\boldsymbol{f}$ = 0,01, $\boldsymbol{g}$ = 0,87, $\boldsymbol{h}$ = 20 y $\boldsymbol{conv}$ = 0,1. Éstos fueron ajustados de acuerdo con una serie de pruebas llevadas a cabo previamente. Como criterio para detener la generación de pulsos, se estableció terminar el proceso cuando el 90 % de neuronas en la red hayan sido activadas. Una vez que se ha detenido la generación de pulsos en la red, son seleccionados los cuatro de mayor entropía, generando así un tensor $S^{480x640x4}$. El cual contiene las imagenes de segmentación.

Las imágenes resultantes del proceso de segmentación propuesto, se presentan en la figura 3.8. En el algoritmo 3.2.3, se presenta el método de segmentación propuesto, donde como entrada se recibe una imagen en el espacio de color RGB y como resultado se obtiene el tensor $S^{480x640x4}$, el cual contiene las imágenes de segmentación.

(a) (b) (c) (d)

Figura 3.8: Resultado del proceso de segmentación.

Algoritmo 3.2.3: Algoritmo de segmentación basado en pulsos de máxima entropía.

Input: I_{rgb}, $f = 0{,}01$, $g = 0{,}87$, $h = 20$

Result: Tensor $S^{480x640x4}$

1 Convertir I_{rgb} a espacio de color HSV $\rightarrow I_{hsv}$;

2 Extraer Hue y aplicar operación de apertura $\rightarrow I_{hue}$;

3 Introducir I_{hue} a red ICM y generar pulsos hasta activar el 90 % de las neuronas $\rightarrow Y_i$;

4 Almacenar los cuatro pulsos de mayor entropía de $Y_i \rightarrow S^{480x640x4}$;

3.2.4. Método basado en regiones de interés (ROI)

Como alternativa a emplear un método de segmentación para poder obtener así los objetos de interés dentro de la imagen, se propuso un método basado en la ubicación semiautomática de regiones de interés (ROI). Las ROI ubican los lugares de estacionamiento dentro de la imagen de captura, es decir, se generan tantas ROI como espacios de estacionamiento se tengan en la imagen.

En algunos trabajos, como en [30] y [5], se lleva a cabo la ubicación de ROI, indicando las coordenadas de cada una dentro de la imagen, demandando tiempo importante para esta tarea. Con el objetivo de disminuir el tiempo de ubicación de ROI, se optó por llevar a cabo la localización de éstas de manera semiautomática, de tal forma que, únicamente se requiera indicar la cantidad de lugares de estacionamiento para ciertas zonas en la imagen. De acuerdo con la distribución de los mismos en el conjunto de imágenes presentado en la sección 3.1, se identificaron dos zonas (superior e inferior), en las cuales se ubican los espacios de estacionamiento, éstas se presentan en la figura 3.9.

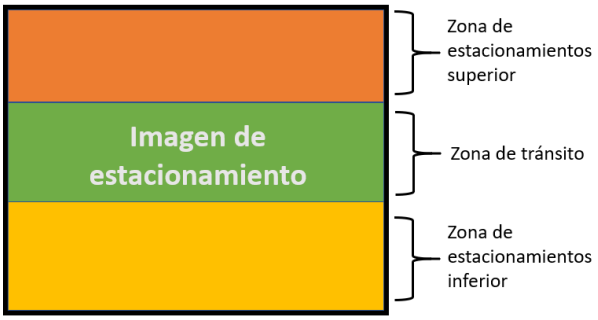


Figura 3.9: Representación de distribución de lugares de estacionamiento en el conjunto de imágenes.

Partiendo de la distribución presentada en la figura 3.9, se diseñó un método para generar las ROI de forma semiautomática, en el cual únicamente se debe indicar la cantidad de lugares de estacionamiento que se encuentran en cada zona, considerando que la cámara de captura no cambie de posición. Este método consiste primeramente en dividir la imagen en tres filas, como se presenta en el inciso (a) de la figura 3.10, las cuales se denominan: fila superior, intermedia e inferior. La altura de las filas superior e inferior corresponde al tamaño en alto del espacio de estacionamiento, mientras que la fila intermedia es el área restante y no es considerada. En seguida, las filas superior e inferior son divididas en columnas, de acuerdo al número de espacios de estacionamiento que se considere pueden ser contenidos dentro de cada fila, de tal manera que, cada columna pueda generar un recuadro que contenga un espacio de estacionamiento. Como resultado, se presenta la imagen del inciso (b) de la figura 3.10, donde la fila inferior se ha dividido

en cuatro columnas y la superior en siete, ya que con ésto se logra contener un lugar de estacionamiento en cada recuadro generado. El número de columnas a dividir, es el parámetro que permite generar las ROI, y éste es establecido de acuerdo al tipo de toma con la que se esté trabajando.

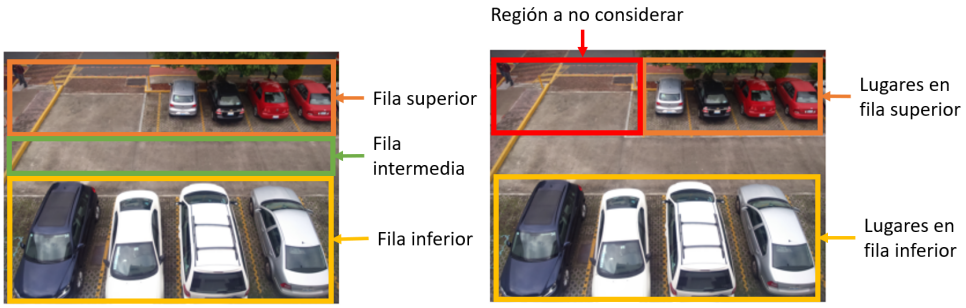


Figura 3.10: Localización de ROI. (a) generación de filas; (b) generación de recuadros (*bounding boxing*).

Una vez hecho lo anterior, dependiendo de la topología del estacionamiento, se puede presentar una región que no se debe considerar, como se muestra en el inciso (a) de la figura 3.11, donde el recuadro en color rojo corresponde a la entrada del estacionamiento, por lo tanto, los recuadros generados dentro de esta región no deben ser tomados en cuenta. Para ésto, se estableció un parámetro, el cual indica si se deben ignorar algunos recuadros generados en cada fila del estacionamiento, que para el caso de la imagen del inciso (a) de la figura 3.11, se establece que los primeros tres recuadros (de izquierda a derecha), de la fila superior se deben saltar, obteniendo como resultado la imagen del inciso (b) de la figura 3.11, la cual muestra las regiones de interés generadas.

Figura 3.11: Resultado de aplicar el método propuesto para la localización de ROI.

En la figura 3.12, se muestra el resultado obtenido del método propuesto para la ubicación semiautomática de regiones de interés (ROI) para cada parte del estacionamiento (centro, lateral, izquierdo y derecho), donde son resaltadas en color rojo las ROI generadas.

(a) (b) (c) (d)

Figura 3.12: Resultado de aplicar el método propuesto para la localización de ROI.

En el algoritmo 3.2.4, se muestran los pasos para la generación de ROI de manera semiautomática, donde se recibe una imagen de un estacionamiento, el tamaño en píxeles de la fila superior T_s, el tamaño en píxeles de la fila inferior T_i, la cantidad de autos contenidos en la fila superior A_s, la cantidad de autos contenidos en la fila inferior A_i, el tamaño de paso para la generación de columnas p y el tipo de estacionamiento EST.

Algoritmo 3.2.4: Algoritmo para generación de ROI.

Input: I_{rgb}, T_s, T_i A_s, A_i, p y EST

Result: ROI

```
1 Dividir I_rgb en tres filas: superior de tamaño T_s, inferior de tamaño T_i e intermedia → F_s, F_i;
2 Dividir las filas superior e inferior en columnas considerando p, de acuerdo con A_s y A_i → ROI;
3 if EST == izquierdo then
4 |   Saltar tres ROI de la parte superior de izquierda a derecha;
5 end
6 if EST == derecho then
7 |   Saltar tres ROI de la parte superior de derecha a izquierda;
8 end
```

3.2.5. Otros métodos de segmentación explorados

Ademas de realizar la segmentación utilizando los modelos de red neuronal pulso-acopladas (PCNN), se exploraron otros métodos para segmentación con el fin de comparar los resultados, y así poder discernir las ventajas y desventajas de cada uno de éstos. Los métodos de segmentación que se exploraron son: el método manual, Otsu y kNN.

El método de segmentación manual

Este método consiste en designar un determinado valor de umbral basado en el análisis del histograma. La imagen se recorre píxel a píxel, y sí se rebasa el valor de umbral entonces el píxel toma un valor 1, de lo contrario se asigna el valor 0; ésto da origen a una nueva imagen con valores de intensidad de 0 y 1 (imagen binaria). A continuación, en la figura 3.13, se presentan los resultados del proceso de segmentación manual llevado a cabo para las imágenes del estacionamiento.

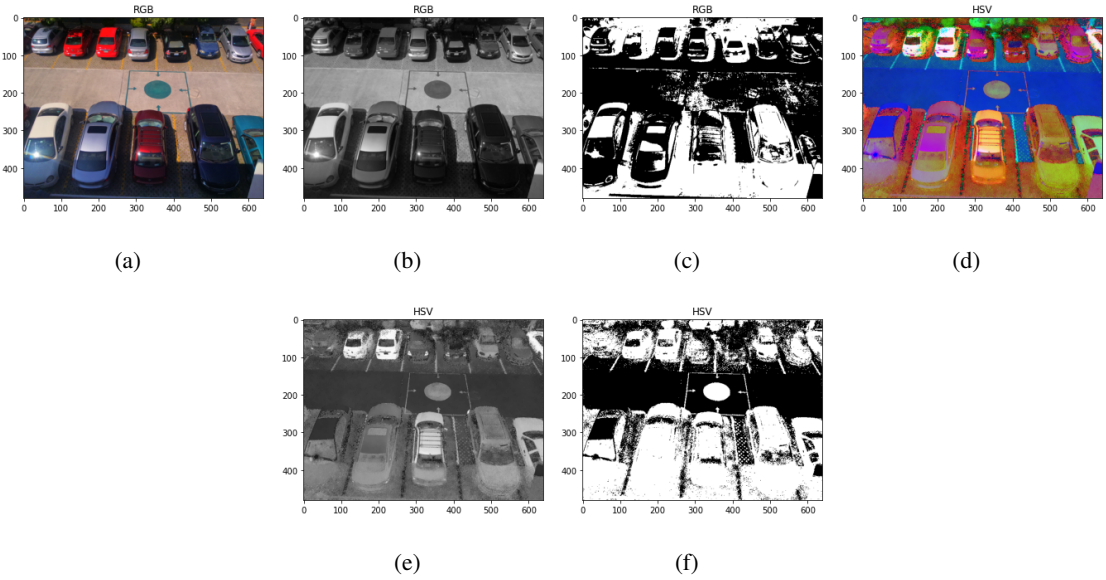


Figura 3.13: Resultados de segmentación manual, (a) imagen en espacio de color RGB, (b) imagen RGB en escala de grises, (c) imagen segmentada, (d) imagen en espacio de color HSV, (e) Imagen HSV en escala de grises, (f) imagen segmentada.

Método de Otsu

Este método fue propuesto por Otsu en [23], y es uno de los más utilizados por su buena respuesta ante imágenes con mala iluminación o con ruido. Este método consiste en encontrar el umbral óptimo de manera automática, esto se hace partiendo del histograma de la imagen, donde se determina la cantidad de niveles de gris en la misma (normalmente entre 0 y 255), los cuales son divididos en dos clases: *C1*, con niveles de gris entre $[0,...,t]$ (siendo t usualmente un valor de 125 para una imagen con 255 valores de intensidad de gris), y *C2* con niveles de gris entre $[t+1,...,255]$. Posteriormente se ajusta el valor de t, de acuerdo al valor de la varianza (σ) en cada clase; el proceso se lleva a cabo de manera iterativa incrementando en uno el valor de t y almacenando el valor de la varianza hasta alcanzar un valor de $t = 255$, al termino de este proceso,

se ubica el umbral óptimo que corresponde a la iteración donde se obtuvo un valor de varianza mayor. En la figura 3.14, se presentan los resultados de segmentación en este método; en el inciso (a) se muestra la imagen original en espacio de color RGB; en el inciso (b), se indica la imagen en escala de grises; en el inciso (c), se muestra el resultado de segmentación por el método de Otsu para una imagen en escala de grises; la imagen del inciso (d) presenta el resultado de conversión al espacio de color HSV; en el inciso (e), se muestra la imagen en HSV convertida a escala de grises; y la imagen del inciso (f), presenta el resultado de segmentación para la imagen HSV en escala de grises.

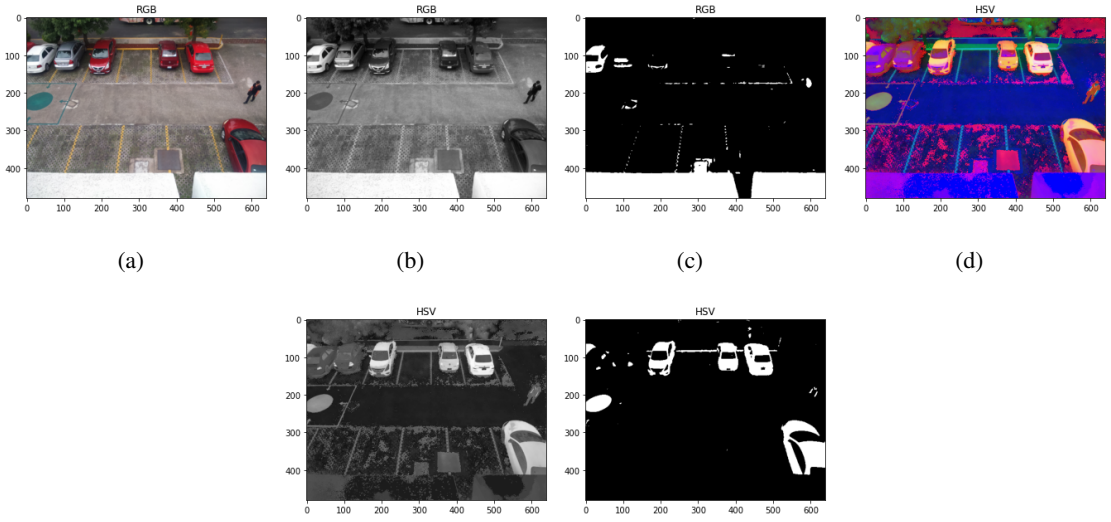


Figura 3.14: Resultados de segmentación por método de Otsu, (a) imagen en espacio de color RGB, (b) imagen RGB en escala de grises, (c) imagen segmentada, (d) imagen en espacio de color HSV, (e) imagen HSV en escala de grises, (f) imagen segmentada.

Segmentación por kNN

Este método de segmentación está basado en el clasificador k vecinos más cercanos (KNN) inicialmente consiste en generar un conjunto de datos, los cuales corresponden a los valores de intensidad de gris de las imágenes que se desea segmentar y sus respectivas etiquetas [33]. En el caso de las imágenes del estacionamiento, se recolectaron muestras del piso a las cuales se les asignó la etiqueta 0, y a las regiones correspondientes a los automóviles se les asignó la etiqueta 1; posteriormente partiendo de las muestras recabadas, se llevó a cabo la segmentación a través de la clasificación de cada píxel dentro de la imagen a segmentar. Para esto se empleó el clasificador KNN mediante el algoritmo 2.3.1. En los incisos del (b) al (f) de la figura 3.15, se muestran los

resultados de la segmentación por *kNN*, teniendo como imagen de entrada la presentada en el inciso (a) de la misma figura.

Con el objetivo de obtener mejores resultados de segmentación por este método, se llevó a cabo un análisis de los datos de aprendizaje, donde se eliminaron los datos menos significativos para evitar efectos de ruido. Las imágenes de los incisos (b) al (e) de la figura 3.15, muestran los resultados obtenidos al emplear los datos de aprendizaje más significados, es decir los de mayor incidencia en las muestras, y la imagen del inciso (f), presenta el resultado de utilizar todos los datos generados para el aprendizaje del clasificador, siendo éste un resultado no adecuado.

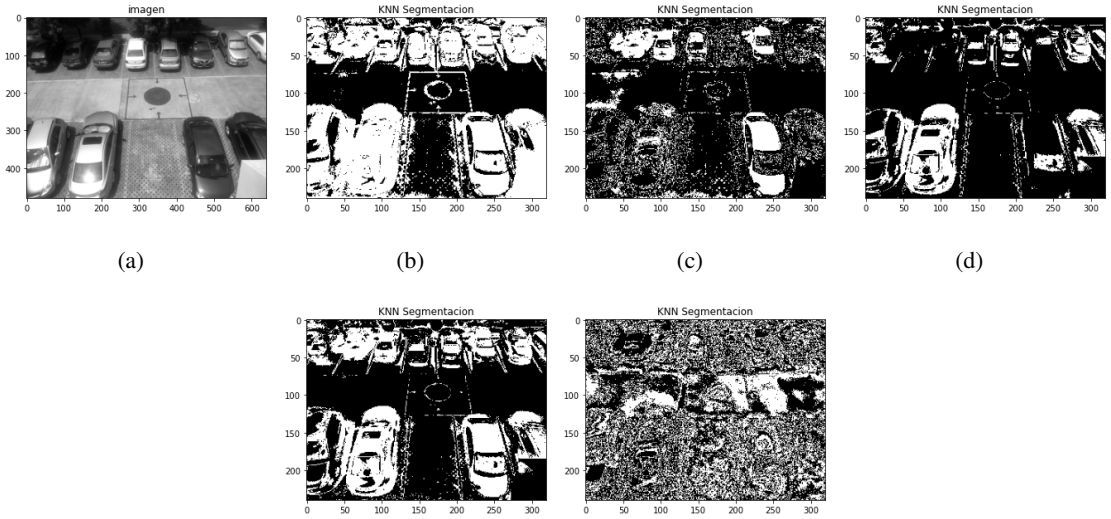


Figura 3.15: Resultados de segmentación por *kNN*; (a) imagen RGB en escala de grises; (b) imagen segmentada tomando el 80 % de los datos más significativos; (c) imagen segmentada tomando en cuenta el 90 % de los datos más significativos; (d) imagen segmentada tomando en cuenta el 50 % de los datos más significativos; (e) imagen segmentada tomando en cuenta el 60 % de los datos más significativos; (f) imagen segmentada utilizando todos los datos para la clasificación.

Comparación de los métodos de segmentación utilizados

De acuerdo con los resultados de segmentación obtenidos con los diferentes métodos explorados, en la tabla 3.1 se presenta una comparativa de estos, donde se describen las ventajas y desventajas observadas durante el proceso de segmentación con las imágenes de estacionamiento presentadas en la sección 3.1.

Método	Ventajas	Desventaja
Manual	Tiempo reducido para su implementación.	El umbral no suele ajustarse para todos los tipos de imagen.
Otsu	Buena respuesta en la mayoría de imágenes en espacio de color HSV.	Presentó un bajo rendimiento en imágenes RGB.
KNN	Buena respuesta para imágenes en espacio de color HSV y RGB.	Demanda un tiempo elevado de procesamiento y memoria para almacenar los datos de aprendizaje.
PCNN	Buena respuesta para imágenes en espacio de color RGB y HSV.	Demanda un costo computacional elevado para implementación. Se requiere de la elección adecuada de sus parámetros
ICM	Respuesta similar al modelo PCNN, reduce el costo computacional y la cantidad de parámetros que demanda el modelo PCNN original.	Es necesario sintonizar de manera adecuada sus parámetros para obtener una buena segmentación.

Tabla 3.1: Comparación de los métodos de segmentación explorados.

3.3. Extracción de características

El proceso de extracción de características se encarga de cuantificar los rasgos de un objeto dentro de una imagen, con el objetivo de que puedan ser interpretadas por una máquina para fines de reconocimiento o clasificación. A continuación, se presentan los métodos de extracción de características utilizados para este trabajo, los cuales son: método basado en la obtención de firma $G[n]$, método basado en la matriz de co-ocurrencia, y el método basado en la obtención de firma $G[n]$ y PCA.

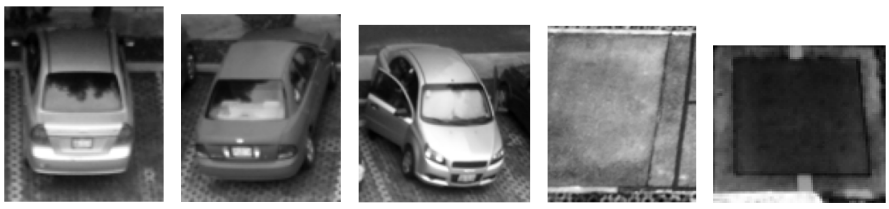
3.3.1. Método basado en la obtención de firma $G[n]$

Además de ser empleada en segmentación, la red ICM también puede ser utilizada para la extracción de características, esto mediante la generación de la firma que produce la red. En [1], [3] y [2], se llevan a cabo trabajos de detección de objetos basados en la firma producida por la red PCNN, donde logran obtener buenos resultados, debido a que la firma generada es invariable al escalado y rotación. La firma que produce la red ICM contabiliza el número de neuronas $Y_{i,j}$ que han sido activadas en una iteración $[n]$ y se representa por $G[n]$ dada por la ecuación 2.13. Un aspecto importante a tomar en cuenta al momento de capturar la firma de un objeto, es la cantidad de pulsos o iteraciones que se deben producir.

Al igual que para el proceso de segmentación, se establecieron los parámetros de la red ICM para obtener la firma del objeto $G[n]$ en base a pruebas, donde se buscó que el patrón de la firma generado por los diferentes automóviles resultara similar, mientras que para los objetos que no corresponden a automóviles presentarán un patrón distinto. Los valores obtenidos son: $\boldsymbol{f} = 0{,}01$, $\boldsymbol{g} = 0{,}8$ y $\boldsymbol{h} = 20$, y la matriz de pesos sinápticos W, como se presenta en la ecuación (3.10). La cantidad de pulsos se estableció a 55, debido a que se observó que a partir del pulso 56 se repite el patrón de la firma, además de esto, se observó que los primeros 15 pulsos siempre se conservan en valor 0, por lo cual se decidió eliminar estos pulsos, y quedarse únicamente con los 40 resultantes, los cuales representan a los objetos.

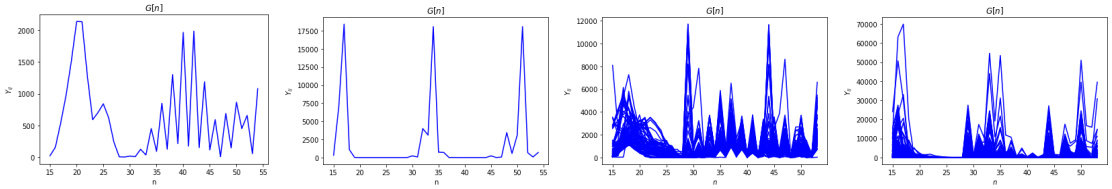
$$W = \begin{pmatrix} 0{,}1 & 0{,}5 & 0{,}1 \\ 0{,}5 & 0 & 0{,}5 \\ 0{,}1 & 0{,}5 & 0{,}1 \end{pmatrix} \tag{3.10}$$

En la figura 3.16 se muestran algunos de los objetos extraídos de las imágenes del estacionamiento después del proceso de segmentación y posterior etiquetado, provenientes del resultado obtenido en la subsección 3.2.2; en el inciso (a), (b) y (c), se muestran algunos automóviles, y en los incisos (d) y (e), se presentan objetos que no pudieron ser eliminados en el proceso de segmentación, los cuales pueden representar partes del estacionamiento como puede ser el techo, el piso o señalamientos. Así mismo en la figura 3.17 se muestra el patrón de las firmas $G[n]$ obtenidas por los autos y los objetos que no representan automóviles; en el inciso (a) se muestra la firma que genera un auto, y en el inciso (b) se muestra el total de firmas obtenidas de los automóviles, mientras que en el inciso (c), se presenta el patrón de firma producido por objetos que no son automóviles, y en el inciso (d), se presenta el total de firmas recabadas por este tipo de objetos.



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

Figura 3.16: Objetos extraídos de las imágenes de un estacionamiento en escala de grises.



(a) (b) (c) (d)

Figura 3.17: Patrón de firmas generado por los objetos en el conjunto de imágenes de entrenamiento; (a) firma generada por un automóvil; (b) firma generada por un objeto distinto a un auto; (c) conjunto de firmas generadas por los automóviles de las imágenes de entrenamiento; (d) conjunto de firmas generadas por objetos distintos a los autos.

3.3.2. Método basado en matriz de co-ocurrencia (GLCM)

En esta etapa se recibe el tensor $S^{480x640x4}$, proveniente del resultado de segmentación obtenido en la subsección 3.2.3. El método que se presenta a continuación, esta basado en el uso de las características de la matriz de co-ocurrencia, geométricas y de color.

Una vez obtenido el tensor $S^{480x640x4}$, se lleva a cabo el proceso de etiquetado en las imágenes dentro del mismo, y se extraen en su forma binaria, en espacio de color RGB y escala de grises. El formato binario permite obtener el área del objeto empleando la ecuación (2.23); la imagen en RGB permite obtener por cada canal, la media de los valores de intensidad; y el formato en escala de grises permite obtener las características de textura a través de la matriz de co-ocurrencia (GLCM), de la cual se extraen los valores de: disimilitud, homogeneidad, correlación, energía, contraste y segundo momento angular (ASM), de acuerdo con las ecuaciones (2.17, 2.18, 2.21, 2.19, 2.16 y 2.20). Las características obtenidas son almacenadas dentro de un vector x, el cual contiene un total de 10 componentes y es representado por la ecuación (3.11).

$$x = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\} \tag{3.11}$$

donde x_1 : disimilitud; x_2 : homogeneidad; x_3 : correlación; x_4 : energía; x_5 : contraste; x_6 : ASM; x_7 : área; x_8 : media del canal R; x_9 : media del canal G; x_{10} : media del canal B.

Una vez que se extraen y almacenan las características dentro del vector x, éstas son normalizadas empleando la función minmax a valores entre -1 y 1, como se muestra en la ecuación (3.12).

$$\text{minmax} = 2\frac{x_i - min}{max - min} \tag{3.12}$$

donde x_i representa el valor que se desea normalizar (que para este caso corresponde a un componente del vector x_i); $\min$ es el valor menor dentro del conjunto de datos al que pertenece x_i, y $\max$ es el valor mayor dentro del conjunto de datos al que pertenece x_i. En la figura 3.18, se muestra la gráfica de las características de textura (homogeneidad y disimilitud) que se recabaron, donde los puntos en color azul corresponden a características de automóviles y los de color rojo a otros objetos, como pueden ser partes del piso del estacionamiento y estructuras externas.

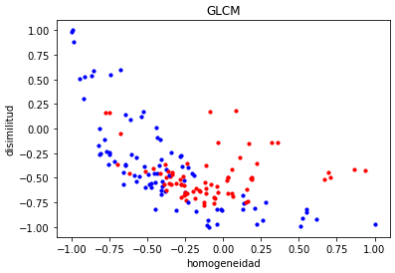


Figura 3.18: Gráfica de características de textura.

3.3.3. Método basado en la obtención de $G[n]$ y PCA

En este método se reciben las ROI generadas en la subsección 3.2.4 para la extracción de sus características. Para lo cual se hace uso de la firma $G[n]$ producida por la red ICM, y para reducir la cantidad de componentes en el vector de características, se emplea el análisis de componentes principales (PCA). A continuación, se exponen los pasos de este método.

El método propuesto recibe una ROI, y ésta es introducida a la red ICM para producir así su firma correspondiente; cabe mencionar que a diferencia del método de extracción de características presentado en la subsección 3.3.1, en este método se hace uso de los canales RGB que componen la ROI con lo cual se puede obtener mayor información del objeto, que en lugar de emplear la imagen en escala de grises. Para obtener los parámetros adecuados de la red ICM, se llevaron a cabo una serie de pruebas con diferentes valores, donde se concluyó que es posible obtener un patrón de firma adecuado con una cantidad de 50 pulsos, empleando el valor de los parámetros: ***f*** = $0{,}1$, ***g*** = $0{,}8$, ***h*** = 20, y W como se presenta en la ecuación (3.13).

$$W = \begin{pmatrix} 0{,}25 & 0{,}5 & 0{,}25 \\ 0{,}5 & 0 & 0{,}5 \\ 0{,}25 & 0{,}5 & 0{,}25 \end{pmatrix} \tag{3.13}$$

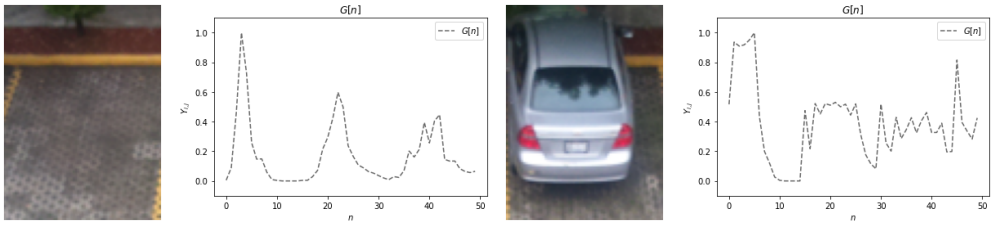
Con los parámetros ajustados en la red ICM, se produce la firma $G[n]$ de acuerdo a la ecuación (2.13), y se introduce dentro de un vector a; éste contiene un total de 50 componentes, los cuales son normalizados a valores entre cero y uno, mediante la ecuación (3.14), generando así el vector

normalizado a_{norm}. Posteriormente se aplica la reducción de dimensionalidad a dos componentes (C_1 y C_2), mediante el método PCA [32], obteniendo un vector x, representado en la ecuación (3.15).

$$a_{norm} = \frac{a}{a_{max}} \tag{3.14}$$

$$x = \{C_1, C_2\} \tag{3.15}$$

En la figura 3.19, se presentan los resultados obtenidos de la extracción de la firma $G[n]$, donde la imagen del inciso (a), corresponde a una ROI de un espacio disponible y la gráfica del inciso (b), presenta su firma respectiva; por otro lado, la imagen del inciso (c), presenta la ROI de un espacio de estacionamiento ocupado por un automóvil, y la gráfica del inciso (d) indica su firma.



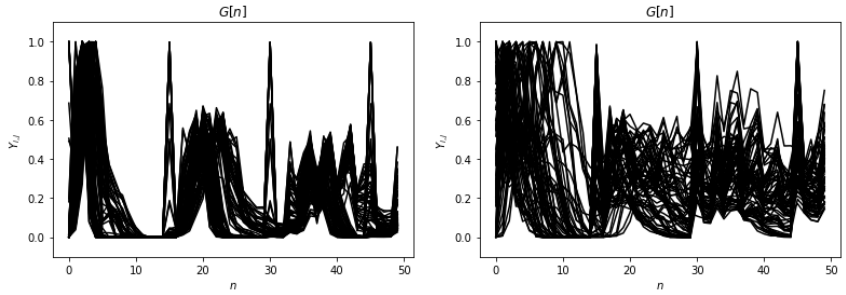
(a) (b) (c) (d)

Figura 3.19: Resultado de extracción de firma $G[n]$.

En las gráficas de la figura 3.20, se presenta el total de firmas recabadas por el conjunto de imágenes de entrenamiento, el cual consta de 90 imágenes, las cuales se emplearon para la generación de conocimiento. Donde la gráfica del inciso (a), muestra el total de firmas obtenidas por las ROI correspondientes a lugares disponibles, mientras que la gráfica del inciso (b), presenta el total de firmas obtenidas por las ROI con lugares ocupados.

Partiendo del conjunto total de firmas generado (ver figura 3.20), se llevó a cabo la reducción de dimensionalidad de las mismas a dos componentes mediante el método PCA. El resultado se presenta en la gráfica de la figura 3.21, donde los puntos en color rojo corresponden a características de espacios ocupados por autos y los puntos en color azul a características de espacios disponibles.

En la tabla 3.2, se presentan los métodos de extracción de características utilizados, donde se muestra el algoritmo de detección de automóviles donde fueron aplicados, así como la cantidad de características generadas.



(a) (b)

Figura 3.20: Conjunto de firmas obtenidas de 90 imágenes; (a) Firmas obtenidas por espacios disponibles; (b) Firmas generadas por espacios ocupados.

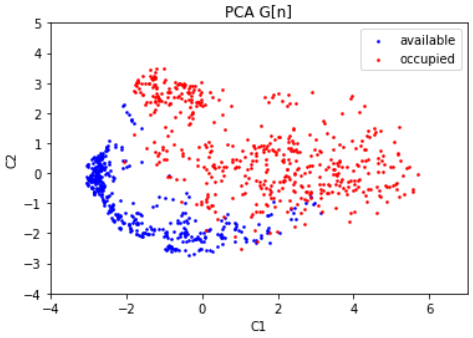


Figura 3.21: Resultados de reducción de componentes.

Método	Área de aplicación
Basado en la obtención de firma $G[n]$	Es empleado en el algoritmo 3.5.2 para detección de autos, donde se aplica a los objetos resultantes del proceso de segmentación, que son extraídos en escala de grises; generando una firma con 55 características.
Basado en matriz de co-ocurrencia	Empleado en el algoritmo 3.5.3, se aplica a los objetos previamente segmentados, que son extraídos en escala de grises, RGB y binario, obteniendo sus características de textura, media de los componentes en RGB y área, generando 10 características.
Basado en la obtención de $G[n]$ y PCA	Empleado en el algoritmo 3.5.4, donde se aplica a las ROI en RGB obtenidas previamente, obteniendo inicialmente su firma con 50 características y posteriormente son reducidas a 2 componentes través de PCA.

Tabla 3.2: Métodos utilizados para la extracción de características.

3.4. Identificación

La etapa de identificación se encarga del reconocimiento de los objetos a través de sus características. En este trabajo se emplearon métodos basados en aprendizaje máquina, donde se llevaron a cabo pruebas con los clasificadores: perceptrón multicapa (MLP), red neuronal de base radial (RBF), k vecinos más cercanos (KNN) y k vecinos más cercanos ponderado (wKNN); donde se descartó el uso de la RBF y wKNN, debido a su bajo nivel de efectividad, empleando las características presentadas en la sección 3.3. Se emplearon métodos de clasificación basados en la red perceptrón multicapa, para la identificación de los objetos dentro de una imagen, y el clasificador KNN, para la identificar el características de iluminación en las imágenes del estacionamiento. A continuación, se presentan los métodos de detección empleados.

3.4.1. Método basado en $G[n]$ y red neuronal perceptrón multicapa

El método que a continuación se describe, recibe las características obtenidas en la subsección 3.3.1, las cuales corresponden a la firma ($G[n]$) para imágenes en escala de grises; dichas características están contenidas en un vector con 41 componentes, mismas que son normalizadas a valores entre -1 y 1.

El clasificador empleado en este método, es la red neuronal perceptrón multicapa ó MLP, el cual puede ser configurado tanto en el número de capas ocultas, como en la cantidad de neuronas en cada una de éstas. Para elegir una arquitectura adecuada a implementar, se decidió llevar a cabo pruebas de validación cruzada con diferentes arquitecturas, y elegir la que obtenga una mejor efectividad. Para ésto se utilizaron un total de 340 muestras, de las cuales 170 corresponden a muestras de firmas generadas por automóviles y 170 de objetos que no son autos; estos se presentan debido a que no fueron eliminados durante el proceso de segmentación, los cuales pueden ser partes del piso y techo del estacionamiento o bien señalamientos que se encuentran dentro del mismo. Para la fase de aprendizaje del clasificador, se estableció el número máximo de épocas a 5500, y una tasa de aprendizaje de $0{,}00025$.

Para el proceso de validación cruzada se implementaron 10 bloques de 34 muestras, y se realizaron pruebas con una arquitectura de red MLP de dos capas ocultas y diferentes configuraciones de neuronas en cada una de estas capas respetando una arquitectura piramidal, se comenzó desde 3 hasta 60 neuronas en la primer capa oculta, y de 2 a 35 neuronas en la segunda capa oculta. Los resultados obtenidos se presentan en la figura 3.22, donde se puede observar que la arquitectura que incorpora 30 neuronas en la primer capa oculta y 20 en la segunda, obtuvo ligeramente una mayor efectividad, la cual fue del 96.4 %, y por lo tanto fue seleccionada para la implementación en la etapa de clasificación. En la figura 3.23, se presenta la gráfica del decremento del error

obtenida del proceso de aprendizaje de la red MLP con la arquitectura de mayor efectividad, así mismo en el inciso (b) de la figura 3.23 se muestra la representación de la misma.

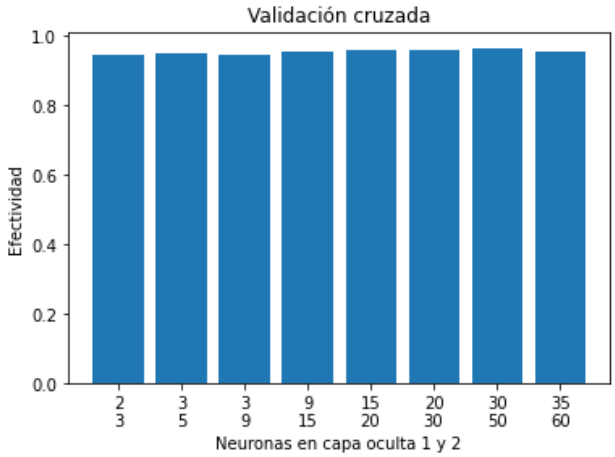
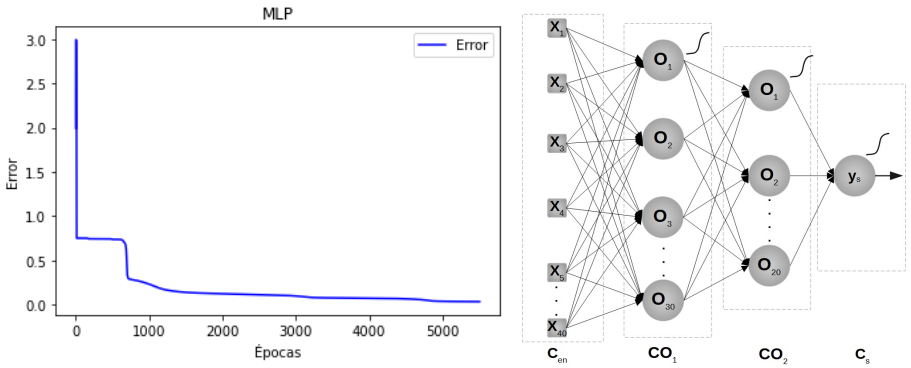


Figura 3.22: Resultados de validación cruzada obtenidos con diferentes arquitecturas de red MLP; el eje x muestra la configuración de neuronas en cada capa oculta, la primer fila muestra la cantidad de neuronas en la capa oculta 2 (CO_2), y la segunda fila presenta las neuronas en la capa oculta 1 (CO_1).



(a) (b)

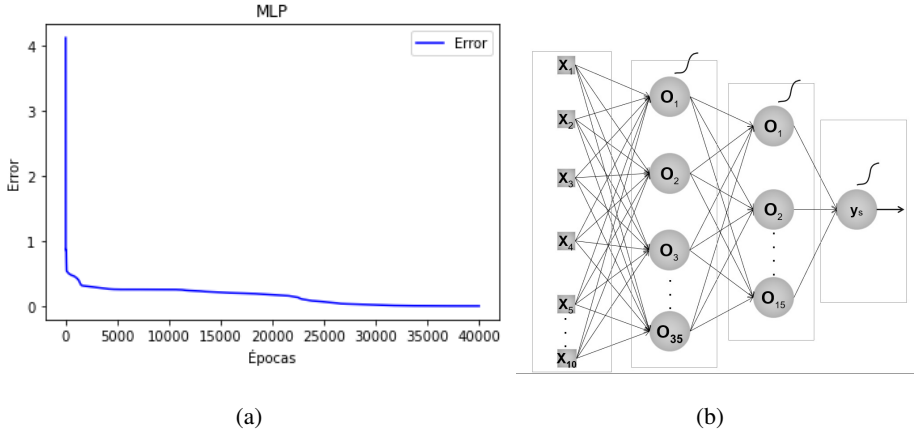
Figura 3.23: Aprendizaje del clasificador MLP; (a) Decremento del error durante el aprendizaje; (b) Topología de red empleada.

3.4.2. Método basado en GLCM, área, color y MLP

El método que se presenta a continuación hace uso de las características obtenidas en la subsección 3.3.2, las cuales corresponden a las características de textura generadas por la matriz de co-ocurrencia, que corresponden a la disimilitud, homogeneidad, correlación, energía, contraste y segundo momento angular, además, se agregan características geométricas y de color, que corresponden al área y la media de intensidad del objeto en los componentes RGB respectivamente. Es decir que, el clasificador va a recibir al vector x normalizado a valores entre -1 y 1 que se

presenta en la ecuación (3.11).

Para este método se empleó el clasificador perceptrón multicapa (MLP), y para su aprendizaje se emplearon 6 imágenes de la parte lateral (ver figura 3.2). De las cuales se extrajeron un total de 160 objetos, donde 74 corresponden a los automóviles y 86 a otros objetos, como pueden ser las estructuras externas y partes del piso del estacionamiento. De lo anterior se generó la base conocimiento mediante la extracción de las características de los 160 objetos, con las que se llevó a cabo el aprendizaje del clasificador. La red MLP se configuró con 35 neuronas en la primer capa oculta y 15 en la segunda, y una sola neurona en la capa de salida (clasificador binario); se estableció un total de 40000 épocas y un error minimo de $0{,}002$. En la figura 3.24, la gráfica del inciso (a) presenta el decremento del error obtenido durante la etapa de aprendizaje y la imagen del inciso (b) presenta la topología de red utilizada en este método.



(a) (b)

Figura 3.24: Aprendizaje del clasificador MLP; (a) Decremento del error durante el aprendizaje; (b) Topología de red empleada.

3.4.3. Método basado en la $G[n]$, PCA y MLP

En este método se hace uso de las características obtenidas en la subsección 3.2.4. Las cuales corresponden a la firma $G[n]$ generada a través de una ROI en RGB, la cual contiene inicialmente un total de 50 pulsos, y posteriormente éstos son reducidos a dos componentes mediante el método PCA, es decir que, para la etapa de clasificación en este método se recibe el vector x obtenido en la ecuación (3.15).

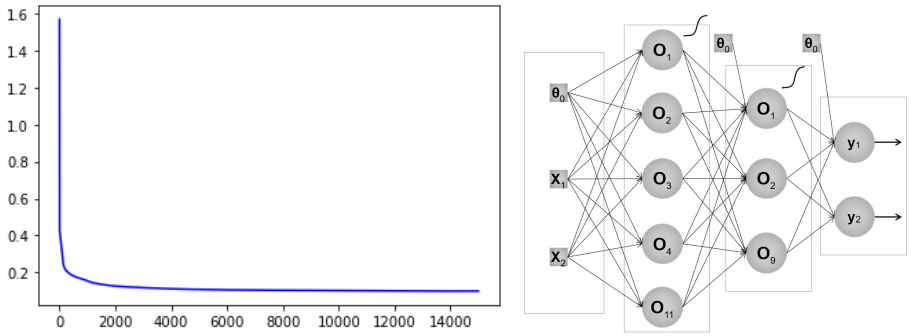
Se empleó el clasificador perceptrón multicapa (MLP), donde la salida de la red es conectada a una función de activación *softmax*, misma que se presenta en la ecuación (3.16).

$$\sigma(y_k) = \frac{e^{y_k}}{\sum_{l=1}^{c} e^{y_1}}, k = 1, ..., c \tag{3.16}$$

donde, σ corresponde a la salida de la red en porcentaje de reconocimiento; y es la salida de una neurona en la capa de salida; k corresponde a la neurona actual; y c es la cantidad de neuronas en la capa de salida.

Se estableció la etiqueta $[0, 1]$, para identificar a las ROI que correspondan a espacios disponibles, y $[0, 1]$ para ocupados. La red MLP se configuró a una arquitectura piramidal, estableciendo dos capas ocultas, con once neuronas en la primer capa, nueve en la segunda y dos en la capa de salida, conectadas a una función de activación *softmax*.

El proceso de aprendizaje se llevó a cabo ajustando la taza de aprendizaje a $0{,}0001$ y el número de épocas a 70000. En la figura 3.25, la gráfica del inciso (a), muestra el decremento del error en la etapa de aprendizaje, y el inciso (b) presenta la topología de red empleada.



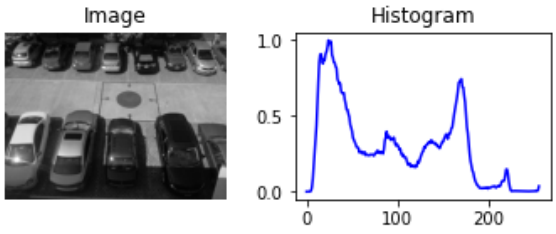
(a) (b)

Figura 3.25: Aprendizaje del clasificador MLP; (a) Decremento del error durante el aprendizaje; (b) Topología de red empleada.

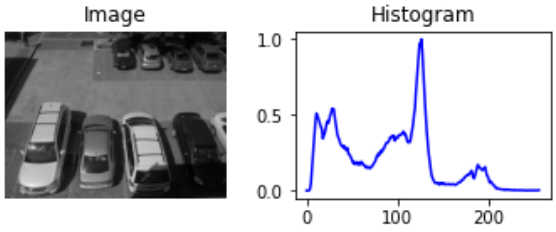
3.4.4. Método basado en el histograma y KNN

El método que se presenta a continuación se empleó para poder clasificar a las imágenes del estacionamiento de acuerdo al tipo de iluminación que presenten; esto para poder emplear el método correspondiente para la detección de lugares disponibles. De acuerdo con el conjunto de imágenes con el que se cuenta (ver sección 3.1), se identificaron tres tipos de imágenes en base a su iluminación. En la figura 3.26, se presentan los tres tipos de iluminación y su respectivo histograma, el cual es generado a través de la imagen en escala de grises; en la imagen del inciso (a), se muestra una imagen con presencia de sombras a causa de una iluminación elevada debido al sol; en la imagen del inciso (b), se presenta una imagen con baja iluminación; y en el inciso (c) se muestra una imagen con iluminación homogénea.

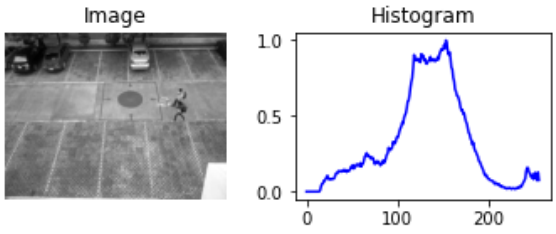
De acuerdo con lo anterior, se llevó a cabo la recolección de 10 histogramas por cada tipo de imagen, generando un total de 30 datos de histograma, con lo cual se generó la base de conoci-



(a)



(b)



(c)

Figura 3.26: Tipos de imagen detectados en base a su iluminación; (a) alta iluminación (presencia de sombras); (b) baja iluminación; (c) iluminación homogénea.

miento para la etapa de aprendizaje del clasificador. En la figura 3.27, se presenta la gráfica de los datos de histograma recolectados, los cuales fueron reducidos a dos componentes mediante el método PCA, donde los puntos en color azul corresponden a características de imágenes con presencia de sombras; los puntos en color rojo corresponden a imágenes con iluminación homogénea; y los puntos en color verde corresponden a imágenes con baja iluminación, que como se pueden ver, las imágenes con baja iluminación y presencia de sombras presentan una separabilidad baja, es decir que su histograma es similar. Para la identificación del tipo de imagen de acuerdo a su iluminación se empleó el clasificador K vecinos más cercanos (KNN), y para su aprendizaje se emplearon las 30 características de histograma previamente recolectadas.

En la tabla 3.3, se presentan los métodos de identificación utilizados, donde se indica el algoritmo de detección de autos donde fueron aplicados, así como el tipo de clasificador empleado y su respectiva configuración.

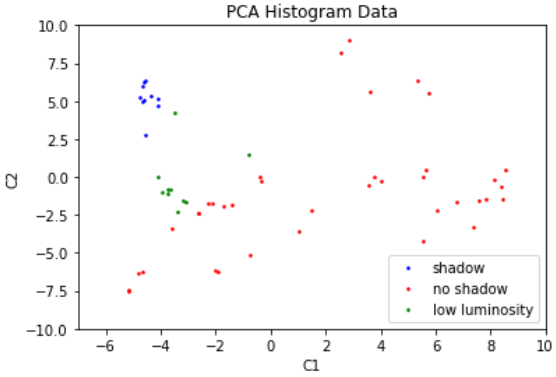


Figura 3.27: Datos de histograma para las imágenes del estacionamiento.

Método	**Área de aplicación**
Basado en $G[n]$ y MLP	Empleado en el algoritmo 3.5.2, se aplica para la identificación de los objetos extraídos en una imagen a través de su firma $G[n]$, para lo cual se emplea la red MLP configurada a una arquitectura piramidal, con 30 neuronas en la primer capa oculta, 20 en la segunda y una neurona en la capa de salida.
Basado en GLCM, área, color y MLP	Se emplea en el algoritmo 3.5.3, donde se aplica para identificar los objetos extraídos en una imagen a través de las características de textura, media de los valores de intensidad en RGB y área, empleando el clasificador MLP configurado a dos capas ocultas, con 35 neuronas en la primer capa oculta y 15 en la segunda, y una neurona en la capa de salida.
Basado en $G[n]$, PCA y MLP	Empleado en el algoritmo 3.5.4 para la identificación de las ROI generadas en una imagen a través de su firma reducida a 2 componentes, empleando para ello el clasificador MLP configurado a una arquitectura piramidal con dos capas ocultas, donde se tienen 11 neuronas en la primer capa oculta y 9 en la segunda, y dos neuronas en la capa de salida las cuales son conectadas a una función softmax.
Basado en la obtención histograma y KNN	Empleado en el algoritmo 3.5.6, se aplica para identificar si una imagen presenta baja iluminación mediante las características de su histograma, para lo cual se emplea el clasificador KNN.

Tabla 3.3: Métodos utilizados para la identificación de objetos.

3.5. Algoritmo general

En las secciones 3.2, 3.3 y 3.4, se presentan métodos para segmentación y ubicación de regiones de interés, extracción de características, y clasificación de objetos (identificación) respectivamente; que para llevar a cabo la detección de espacios disponibles en un estacionamiento, se han relacionando algunos de estos. A continuación, se presentan los algoritmos que integran las diferentes técnicas de segmentación, ubicación de ROI, extracción de características y clasificación presentados previamente para la detección de espacios disponibles en un estacionamiento, así como, la integración de estos en un algoritmo general, con el cual sea posible llevar a cabo la detección de espacios disponibles, en las diferentes topologías de estacionamiento y condiciones de iluminación, con las que se cuentan en el conjunto de imágenes presentado en la sección 3.1.

Inicialmente se diseñó el algoritmo 3.5.1, con el cual se lleva a cabo la ubicación de las zonas donde se encuentran los automóviles, para posteriormente dar lugar al proceso de segmentación. Para la detección de autos en el estacionamiento, se toma la imagen segmentada, y se considera como estructuras de un auto a las regiones con intensidad de 1, por lo cual este método no incorpora las etapas de extracción de características e identificación. De acuerdo con los resultados obtenidos, los cuales son presentados en la sección 4.1, el método presentó una correcta detección de los automóviles, sin embargo, en imágenes con presencia de sombras, se detectan dos o mas autos como uno solo, debido a la unión de estructuras ocasionado por la sombra. A continuación se presenta el algoritmo propuesto, el cual recibe una imagen en el espacio de color RGB y como resultado se obtiene la misma imagen pero con la detección de los automóviles.

Algoritmo 3.5.1: Detección de automóviles mediante segmentación por zonas.

Input: I_{rgb}

Result: Imagen con detección de autos (I_d)

1 Obtener zonas segmentadas mediante el algoritmo 3.2.1 $\rightarrow S_a$, S_b;
2 Aplicar operaciones de apertura y cierre a S_a, S_b;
3 Etiquetar los objetos contenidos en $S_a, S_b \rightarrow E_a, E_b$;
4 **for** *Cada etiqueta en* E_a *y* E_b **do**
5 | almacenar coordenadas x,y del objeto;
6 | Dibujar las coordenadas del objeto en la imagen original $I_{rgb} \rightarrow I_d$
7 **end**

Uno de los principales inconvenientes que presenta el algoritmo 3.5.1, es la susceptibilidad al movimiento de la cámara de captura; ya que ocasiona un desajuste en la ubicación de las zonas de interés. Para contrarrestar lo anterior, se diseñó el algoritmo 3.5.2, mismo que hace uso de los métodos presentados en las subsecciones 3.2.2, 3.3.1 y 3.4.1, para segmentación, extracción

de características y clasificación respectivamente; donde se elimina la etapa de ubicación de las zonas donde se encuentran los lugares de estacionamiento, dejando que el proceso de segmentación realice la ubicación de los objetos de interés, mediante la separación de éstos con el fondo. Este método se empleó para la detección de autos en las imágenes de la parte centro, derecha e izquierda del estacionamiento. De acuerdo con los resultados obtenidos, mismos que son presentados en la sección 4.2; se presentó una adecuada detección de los automóviles en imágenes con iluminación homogénea, o bien con ligeras variaciones de iluminación, sin requerir de la ubicación de zonas de interés, sin embargo, en imágenes con presencia de sombras, las cuales son generadas a partir de la luz del sol, la efectividad disminuye, ya que se genera la unión de estructuras debido a ésto. A continuación, se presenta el algoritmo, el cual recibe una imagen en el espacio de color RGB, y como resultado se obtiene la misma imagen con la ubicación de los autos detectados, encerrándolos en un recuadro de color rojo.

Algoritmo 3.5.2: Detección de autos en estacionamiento mediante $G[n]$ y MLP.

Input: I_{rgb}

Result: Imagen con ubicación de los automóviles I_d

```
Convertir I_rgb a espacio de color HSV → I_hsv;
Extraer Hue y aplicar operación de apertura → I_hue;
Aplicar segmentación con ICM mediante algoritmo 3.2.2 → I_b;
Etiquetar imagen I_b → I_E;
for cada objeto en I_E do
    Obtener las coordenadas del objeto → [x, y];
    Acceder al objeto en escala de grises → I_obj;
    Generar firma de I_obj mediante método de subsección 3.3.1 → G[n];
    Ingresar G[n] a la red MLP configurado de acuerdo al método de la subsección 3.4.3 → y_s;
    if y_s == 1 then
        Es un auto, indicarlo en I_rgb mediante sus coordenadas → I_d;
    end
end
```

Al realizar pruebas con las imágenes de la parte lateral del estacionamiento con el algoritmo 3.5.2, no se obtuvo un desempeño adecuado, ésto a causa principalmente de un menor tamaño en los automóviles, consecuencia de una mayor distancia de la cámara de captura; lo cual hace que los autos pequeños sean removidos durante los procesos de pre-procesamiento, segmentación y etiquetado. Debido a lo anterior, se llevó a cabo un ajuste en el proceso de segmentación con la red ICM, para evitar que las estructuras de menor tamaño fueran removidas, además para el reconocimiento de los objetos se emplearon características de textura en lugar de la firma. El algoritmo propuesto hace uso de los métodos presentados en las subsecciones 3.2.3, 3.3.2 y 3.4.2,

para llevar a cabo la segmentación, extracción de características y clasificación respectivamente. Los resultados obtenidos de este método, los cuales son presentados en la sección 4.3, muestran una detección adecuada de los autos, incluyendo los de menor tamaño, sin embargo, en algunos casos los automóviles de menor tamaño no fue posible detectarlos debido a cambios en la iluminación, además, se observó que las estructuras de los autos con tonalidad similar al piso de estacionamiento son eliminados en el proceso de segmentación. A continuación, en el algoritmo 3.5.3, se presenta el método propuesto, donde se recibe una imagen en RGB y como resultado se obtiene la ubicación de los automóviles detectados, remarcandolos en color rojo dentro de la misma imagen.

Algoritmo 3.5.3: Detección de autos en estacionamiento mediante GLCM y MLP.

Input: I_{rgb}

Result: Imagen con ubicación de los automóviles I_d

```
Convertir I_rgb a espacio de color HSV → I_hsv;
Extraer Hue y aplicar operación de apertura → I_hue;
Aplicar segmentación con ICM mediante algoritmo 3.2.3 → S^{480x640x4};
Etiquetar imágenes en S^{480x640x4} → I_E^{480x640x4};
for cada imagen en I_E^{480x640x4} do
    for cada objeto en I_E do
        Obtener las coordenadas del objeto → [x, y];
        Acceder al objeto en escala de grises, binario y RGB → I_gray, I_b, I_rgb;
        Obtener características mediante el método de la subsección 3.3.2 → x;
        Ingresar x a la red MLP empleando el método de la subsección 3.4.2 → y_s;
        if y_s == 1 then
            Es un auto, indicarlo en I_rgb mediante sus coordenadas → I_d;
        end
    end
end
```

Los algoritmos 3.5.2 y 3.5.3 presentados anteriormente, realizan una adecuada detección de autos en las imágenes de estacionamiento con las que se probaron (ver resultados en las subsecciones 4.2 y 4.3), sin embargo, su rendimiento disminuye cuando se presentan efectos de sombra en las imágenes, lo que ocasiona la unión de estructuras, y que en consecuencia hace que se puedan detectar dos o más objetos como uno. De acuerdo con lo anterior, se optó por emplear un método alternativo, con el cual sea posible llevar a cabo la detección de espacios disponibles y ocupados en las imágenes del estacionamiento bajo los efectos de sombra. Para ésto, se emplearon los métodos presentados en las subsecciónes 3.2.4, 3.3.3 y 3.4.3; los cuales conforman el algoritmo 3.5.4, donde se sustituye la etapa de segmentación por la generación semiautomática

de regiones de interés (ROI). Los resultados presentados en la sección 4.4, fueron obtenidos de acuerdo a una serie pruebas llevadas a cabo con imágenes de las partes centro, derecha, izquierda y lateral del estacionamiento; obteniendo un porcentaje de efectividad adecuado. El principal inconveniente de este método, es la susceptibilidad a los cambios de posición de la cámara de captura, ya que ésto provoca una mala ubicación de las ROI. A continuación, se presenta el algoritmo propuesto, donde se recibe una imagen de un estacionamiento en RGB I_{rgb}, el tamaño en píxeles de la fila superior T_s, el tamaño en píxeles de la fila inferior T_i, la cantidad de autos contenidos en la fila superior A_s, la cantidad de autos contenidos en la fila inferior A_i, el tamaño de paso para la generación de columnas p y el tipo de estacionamiento EST.

Algoritmo 3.5.4: Detección de espacios disponibles mediante ROI, $G[n]$ y PCA.

Input: I_{rgb}, T_s, T_i, A_s, A_i, p, EST

Result: Imagen con ubicación de los espacios disponibles y ocupados I_d

```
Ubicar las ROI de acuerdo al algoritmo 3.2.4 → ROI;
for cada ROI do
    Obtener G[n] mediante el método de la subsección 3.3.3→ a;
    Normalizar a → a_norm;
    Reducir a_norm a dos componentes → x;
    Identificar x por MLP empleando el método de la subsección 3.4.1 → σ;
    if σ [0,1] >80 % then
        Marcar ROI como ocupado;
    end
    if σ [1,0] >80 % then
        Marcar ROI como disponible;
    end
end
```

De acuerdo con los resultados obtenidos en los algoritmos propuestos, se propuso la integración de estos en un algoritmo general, es decir que utilice los algoritmos 3.5.2, 3.5.3 y 3.5.4, con algunas modificaciones para mejorar su desempeño, y tener así, un método para llevar a cabo la detección de espacios disponibles en las cuatro diferentes partes del estacionamiento (centro, derecho, izquierdo y lateral). Partiendo del algoritmo 3.5.2, con el cual se obtuvo mayor efectividad (ver tabla 4.6), en comparación con los algoritmos 3.5.1, 3.5.3 y 3.5.4; se realizó la combinación de éste con el método propuesto en el algoritmo 3.5.3, generando así el algoritmo 3.5.5, donde la etapa de procesamiento es similar al método presentado en 3.5.3; la etapa de segmentación, continúa con la selección de las cuatro imágenes de mayor entropía, pero ahora empleando el método de segmentación propuesto en la subsección 3.2.2, ésto debido a que el método de segmentación empleado en 3.5.3, genera mayor cantidad de objetos que suelen ser irrelevantes para

la identificación de los automóviles, además, demanda mayor tiempo de ejecución; por otra parte, los parámetros de la red ICM fueron establecidos de manera local, es decir, por cada tipo de imagen de estacionamiento, teniendo así cuatro diferentes valores para cada parámetro en la red, es por esto que, el algoritmo 3.5.5 presenta la entrada $Parte$, para indicarle el tipo de estacionamiento en el cual se está operando. Para el reconocimiento de los objetos se hace uso de la firma presentada en la subsección 3.3.1, pero en esta ocasión se obtiene para los canales RGB del objeto. Y para la clasificación, se hace uso de el método presentado en la subsección 3.4.2. En base a los resultados obtenidos del algoritmo 3.5.5, los cuales son presentados en la sección 4.5; éste método es capaz de llevar a cabo la detección adecuada de los automóviles en las imágenes de la parte centro, derecho, izquierdo y lateral, mejorando la efectividad conseguida con los métodos propuestos en los algoritmos 3.5.1, 3.5.3 y 3.5.4.

Algoritmo 3.5.5: Combinación de algoritmos 3.5.2 y 3.5.3.

Input: I_{rgb}, $Parte$

Result: Imagen con ubicación de los automóviles I_d

```
1  Convertir I_rgb a espacio de color HSV → I_hsv;
2  Extraer Hue y aplicar operación de apertura → I_hue;
3  Aplicar segmentación con ICM mediante algoritmo 3.2.2 → S^{480x640x4};
4  Etiquetar imágenes en S^{480x640x4} → I_E^{480x640x4};
5  for cada imagen en I_E^{480x640x4} do
6      for cada objeto en I_E do
7          Obtener las coordenadas del objeto → [x, y];
8          Acceder al objeto en RGB → I_rgb;
9          Obtener G[n] mediante el método de la subsección 3.3.1 → x;
10         Ingresar x a la red MLP, mediante el método de la subsección 3.4.2 → y_s;
11         if y_s == 1 then
12             Es un auto, indicarlo en I_rgb mediante sus coordenadas → I_d;
13         end
14     end
15 end
```

El inconveniente principal del método presentado anteriormente (algoritmo 3.5.5), es la susceptibilidad a los efectos de sombra, ya que estos pueden generar que las estructuras de los autos se unan. Para solucionar este inconveniente se decidió incluir en el algoritmo general, el método basado en la generación semiautomática de ROI (presentado en el algoritmo 3.5.4), ya que éste presentó mejores resultados en imágenes con presencia de sombras y baja iluminación. Es decir, el algoritmo principal está compuesto por los algoritmos 3.5.5 y 3.5.4, los cuales serán empleados de acuerdo al tipo de imagen de entrada, esto es, cuando la imagen de entrada presente baja

iluminación o efectos de sombra, entonces se hace uso del algoritmo 3.5.4, y cuando la imagen de entrada presente una iluminación homogénea, entonces se emplea el algoritmo 3.5.5. Para el reconocimiento del tipo de imagen de entrada se empleó el análisis del histograma, y para la clasificación de éstas se hace uso del clasificador KNN, de acuerdo con el método presentado en la subsección 3.4.4.

El método general para la detección de autos en un estacionamiento, se presenta en el algoritmo 3.5.6, donde se recibe una imagen en RGB y se indica el tipo de estacionamiento donde se está operando, y como resultado se obtiene la misma imagen con la detección de los espacios disponibles.

Algoritmo 3.5.6: Detección general de espacios disponibles.

Input: I_{rgb}, $Parte$

Result: Imagen con espacios disponibles (I_d)

```
Obtener histograma de I_rgb → h;
Identificar tipo de imagen por KNN empleando el método 3.4.4 → u;
if u == 0 then
    Emplear algoritmo 3.5.4 → I_d;
end
if u == 1 then
    Emplear algoritmo 3.5.5 → I_d;
end
```

3.6. Interfaz de usuario

En esta sección se aborda el desarrollo de la interfaz de usuario, la cual tiene como función principal, indicar el porcentaje de espacios disponibles y ocupados de un estacionamiento, a través de la implementación del algoritmo general propuesto en la sección 3.5. Para desarrollar la interfaz de usuario se utilizó la biblioteca *Tkinter*. A continuación, se presenta la interfaz de usuario diseñada, la cual implementa en su núcleo el algoritmo 3.5.6, propuesto para la detección de automóviles.

La interfaz desarrollada implementa una serie de controles e indicadores para permitir al usuario una fácil interacción con las diferentes funciones. En la figura 3.28 se presenta la ventana de inicio, la cual como ya se mencionó, cuenta con una serie de controles e indicadores posicionados en la parte izquierda, y en la parte derecha cuenta con una ventana de visualización, donde se muestra en un inicio una imagen por defecto, y al momento de cargar una imagen de un estacionamiento o abrir la cámara, ésta será reemplazada por la nueva imagen. A continuación, se

describe el modo de operación de la interfaz, la cual se logró desarrollar para funcionar mediante la selección de una imagen que se encuentre dentro de los archivos de la computadora, y también para la captura de video.

Figura 3.28: Ventana de inicio de la interfaz de usuario.

En la figura 3.29 se presentan las pestañas de los dos modos de operación de la interfaz, los cuales son: imagen y video. En el modo imagen, se permite al usuario cargar una imagen en formato *.jpg* desde una ruta de archivos, y posteriormente llevar a cabo la detección de automóviles mediante el algoritmo general, donde se indicará la cantidad de autos detectados y la ubicación de éstos en la imagen. El modo de video, permite abrir la cámara web y a través de ésta lleva a cabo la captura de video; por cada cuadro capturado se aplica el algoritmo general, obteniendo así la ubicación de los autos detectados y la cantidad de éstos; cabe mencionar que, cuando ya no se desee emplear el modo de video, es necesario oprimir el botón *"parar video"*, para poder visualizar nuevamente las imágenes que sean cargadas desde el gestor de archivos.

Figura 3.29: Modos de operación de la interfaz de usuario y sus respectivos controles e indicadores

Algunas de las funciones de los controles que se encuentran dentro de los modos de operación son los siguientes:

- Modo imagen:
  - *Abrir imagen*, botón que permite al usuario abrir una imagen en formato .jpg desde el

gestor de archivos.

- *Detectar*, botón que ejecuta un algoritmo para la detección de espacios disponibles.
- *Autos detectados*, es un indicador que muestra la cantidad de autos detectados en la imagen.
- *Cerrar*, botón para cerrar la interfaz.

- Modo video:

  - *Abrir cámara*, botón que permite abrir la cámara web de la computadora para llevar a cabo la detección de autos mediante la captura de video.
  - *Autos detectados*, indicador que muestra la cantidad de autos detectados en cada cuadro capturado.
  - *Cerrar*, botón que permite cerrar la interfaz.

El funcionamiento de la interfaz de usuario se presenta en el diagrama de la figura 3.30, donde se inicia la selección del modo de operación; una vez hecho lo anterior se continúa introduciendo la imagen proveniente del gestor de archivos o de la cámara, según sea el caso, y ésta es introducida al algoritmo general, el cual regresa la imagen resultado, que contiene la ubicación de los automóviles, así como la cantidad de los mismos. En el caso de emplear el modo de video, se deben tener en cuenta dos aspectos importantes, éstos son: la tasa de refresco o tiempo de captura y visualización de un nuevo cuadro, y una vez que se desee cambiar a modo imagen, se debe pulsar el botón *"parar video"*, para que sea posible visualizar nuevamente una imagen cargada desde el gestor de archivos.

En la figura 3.31, se presenta la secuencia para cargar una imagen desde el gestor de archivos. En el inciso (a), se presenta la captura de pantalla una vez que se ha presionado el ícono *abrir una imagen*; esta acción abre la ventana del explorador de archivos, para permitirle al usuario seleccionar una imagen en formato *.jpg*; en el inciso (b) se muestra la interfaz de usuario donde la ventana de visualización muestra la imagen seleccionada previamente, así mismo, el contador de autos se inicializa en valor cero.

De igual manera en la figura 3.32, se presenta la secuencia para emplear el modo de video; en el inciso (a), se muestra la habilitación del panel *"video"*; el inciso (b), muestra el video obtenido una vez que se ha presionado el botón *äbrir cámara¨*; el inciso (c), muestra el resultado de oprimir el botón *"parar video"*, el cual cierra la conexión con la cámara web, y posteriormente establece la imagen por defecto en el panel de visualización.

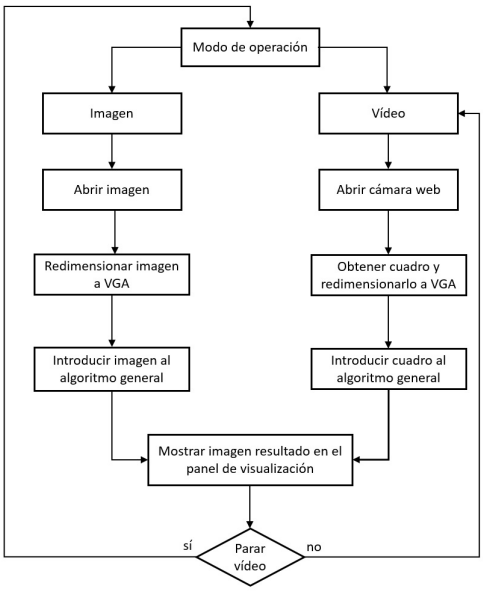
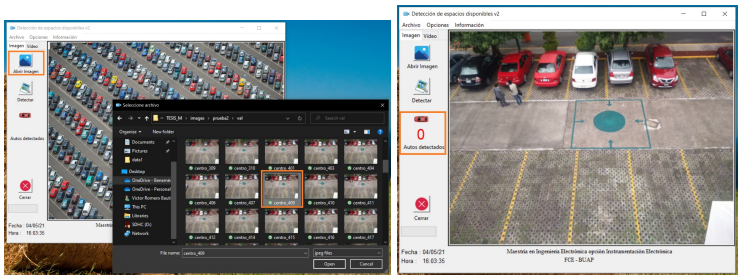
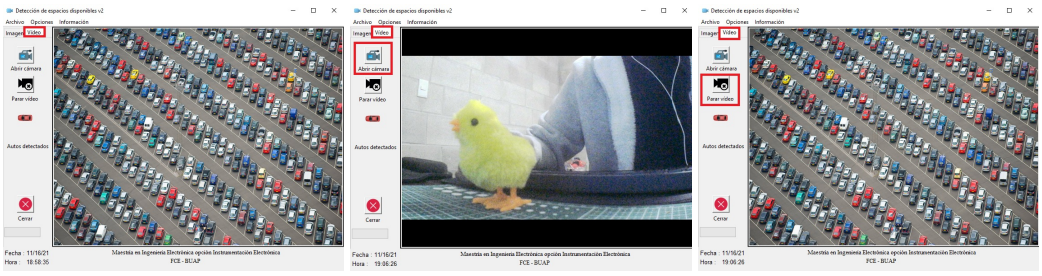


Figura 3.30: Diagrama del funcionamiento de la interfaz de usuario.



(a) (b)

Figura 3.31: Modo de operación por imagen, (a) abrir una imagen desde el explorador de archivos; (b) la imagen seleccionada se muestra en la ventana de visualización.

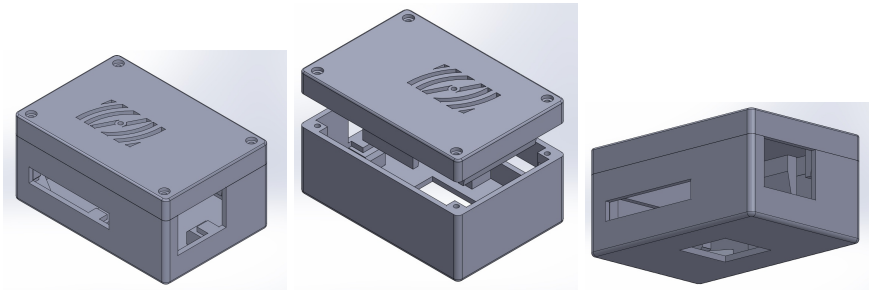


(a) (b) (c)

Figura 3.32: Modo de operación por captura de video, (a) panel para video; (b) cuadro capturado; (c) resultado de oprimir botón *"parar video"*.

3.7. Implementación en Raspberry Pi

En esta sección se presenta el proceso para la implementación de la interfaz de usuario en conjunto con el algoritmo general, en la tarjeta *Raspberry Pi*. El modelo de tarjeta utilizado para este trabajo, es el *Raspberry Pi 4* en la versión de 4 GB de memoria RAM, a la cual se le cargó la versión de escritorio del sistema operativo *Raspbian* de 32 bits. Éste SO incluye el interprete de *Python* en las versiones 2 y 3. En este trabajo se hace uso de la versión $3{,}7$ para la implementación de los algoritmos. Además, se instalaron las bibliotecas: *numpy*, *pandas*, *Pillow*, *opencv* y *skimage*. Por otro lado, se llevó a cabo el diseño de una cubierta para poder introducir dentro de ésta a la tarjeta *Raspberry Pi* y la cámara para captura de video; en la figura 3.33, se presenta el diseño realizado en el software de diseño asistido por computadora (CAD) *SolidWorks*, donde se muestra mediante una vista isométrica la cubierta diseñada; siendo la imagen del inciso (a), la vista inferior; el inciso (b), presenta una vista explosionada; y la imagen del inciso (c), es la vista superior. La cubierta consiste de dos piezas: la base principal y su tapa. Posteriormente de haber llevado a cabo el diseño, se continuó con la fabricación de las piezas que la componen mediante impresión 3D, y para unir la base principal con la tapa, se hace uso de tornillos M3x20.

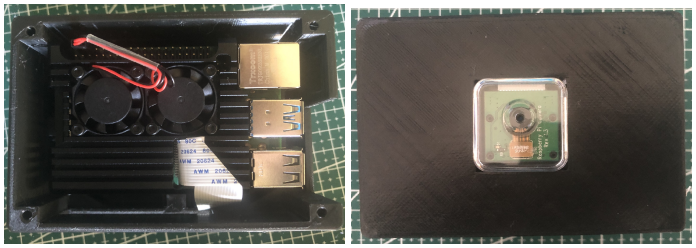


(a) (b) (c)

Figura 3.33: Diseño de la cubierta para la tarjeta *Raspberry Pi* y la cámara.

En la figura 3.34, se presenta a la tarjeta *Raspberry Pi* y la cámara de captura dentro de la cubierta diseñada; en la imagen del inciso (a), se muestra la vista inferior, donde se puede apreciar la ubicación de la tarjeta *Raspberry Pi*; en el inciso (b), se presenta la vista superior, en la cual se ha colocado la cámara. A continuación, se presenta el proceso de implementación de la interfaz de usuario en esta tarjeta, la cual a su vez ejecuta el algoritmo 3.5.6, para llevar a cabo la detección de lugares disponibles.

Además de la instalación de las bibliotecas mencionadas anteriormente, para ejecutar adecuadamente el programa principal, fue necesario inhabilitar el uso de iconos en la ventana de la interfaz. En la figura 3.35 se presenta la ejecución de la interfaz de usuario en la tarjeta *Raspberry Pi*, donde la salida de video de ésta es conectada directamente a un monitor, en el cual se puede



(a) (b)

Figura 3.34: *Raspberry Pi* y cámara dentro dentro de la cubierta.

observar la ejecución del programa principal, el cual a su vez presenta la interfaz de usuario.

Figura 3.35: Ejecución de la interfaz de usuario en la tarjeta *Raspberry Pi*.

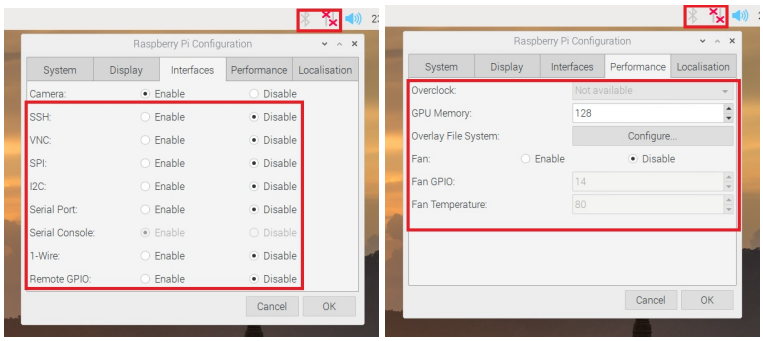
Una vez ejecutada la interfaz de usuario, se realizaron las pruebas de su funcionamiento, sobre todo para medir la velocidad de ejecución del algoritmo general. En la figura 3.36, se muestran el uso de la interfaz de usuario, en modo de operación por imagen, donde se puede observar la imagen que se ha cargado.

De igual manera, se llevaron a cabo pruebas con el modo de video, sin embargo, debido a que no se cuenta con acceso al estacionamiento, no fue posible llevar a cabo las pruebas de detección en es modo de operación. A pesar de esto, se llevaron a cabo pruebas para identificar posibles congelamientos en la captura de video; donde por un lapso de 20 minutos no se detectó algún problema de congelamiento, considerando que el tiempo de refresco entre un cuadro y otro oscila entre 2 y 4 segundos.

Así mismo con el objetivo de reducir la cantidad de procesos que ejecuta el sistema operativo *Raspbian*, se desactivaron las interfaces que incorpora esta tarjeta, exceptuando el uso de la cámara, en la figura 3.38, se muestran las interfaces desactivadas. En la sección 4.7 del capítulo 4 se presentan los resultados obtenidos de la implementación del algoritmo general en tarjeta *Raspberry Pi*.

Figura 3.36: Ejecución del modo de operación por imagen.

Figura 3.37: Ejecución del modo de captura de video a través de la cámara web.



(a) (b)

Figura 3.38: Configuración del sistema operativo *Raspbian* para reducir el costo procesamiento de otras tareas que no se requieren para la aplicación.

Capítulo 4

Resultados

En este capítulo se presentan los resultados de los algoritmos desarrollados, los cuales fueron generados a partir de los métodos de segmentación, extracción de características e identificación presentados en la secciones 3.2 ,3.3 y 3.4. Se utilizó la efectividad y precisión, para evaluar los algoritmos propuestos; en las ecuaciones (4.1) y (4.2), se presenta la relación matemática de éstas, donde VP corresponde a los verdaderos positivos, FP indica los falsos positivos y M es la cantidad de muestras sobre las cuales se efectúa la evaluación [21].

$$\text{Efectividad} = \frac{VP+FP}{M} \tag{4.1}$$

$$\text{Precisión} = \frac{VP}{VP+FP} \tag{4.2}$$

Además, se muestran los resultados obtenidos en la implementación de los algoritmos en la tarjeta *Raspberry Pi 4*.

4.1. Resultados de algoritmo 3.5.1

El método que se presenta a continuación, esta basado en el algoritmo de segmentación 3.2.1, el cual es presentado en la subsección 3.2.1. En el algoritmo 3.5.1, se presenta el método propuesto para llevar a cabo la detección de los automóviles en un estacionamiento mediante la segmentación basada en zonas. Donde inicialmente se recibe una imagen en el espacio de color RGB, y como resultado regresa la misma imagen de entrada, pero con la ubicación de los automóviles, generando un recuadro sobre éstos (*bounding boxing*). Este método identifica como un auto a los objetos dentro de la imagen segmentada con niveles de intensidad en valor 1.

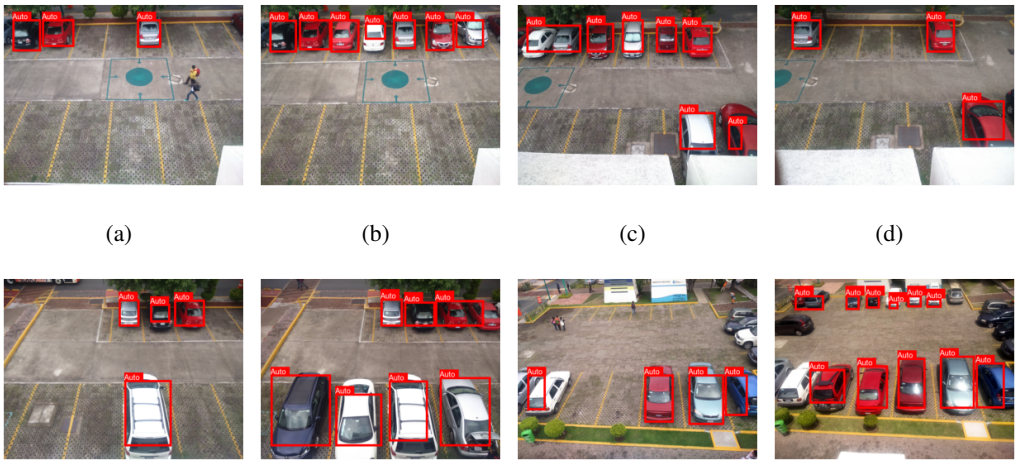
Se llevaron a cabo pruebas con el algoritmo 3.5.1, empleando 82 imágenes de un estacionamiento, el cual está compuesto de cuatro tomas diferentes: centro, derecha, izquierda y lateral. En la tabla 4.1, se presenta el porcentaje de efectividad promedio obtenido de las pruebas realizadas para las diferentes tomas del estacionamiento. Donde se logró obtener un 100 % de efectividad

en imágenes con iluminación homogénea, mismas que son producidas en clima nublado, por otro lado, la efectividad disminuyó a 75% y 60% debido a dos factores, el primero se debe a cambios de iluminación ocasionados por la luz del sol, y el segundo factor es el movimiento de la cámara de captura, que provoca una mala ubicación de la zona de interés.

Toma del estacionamiento	Imágenes de prueba	Efectividad con clima nublado	Efectividad con clima soleado
Derecha	31	100%	80%
Centro	30	80%	75%
Izquierda	7	75%	80%
Lateral	4	60%	N/A

Tabla 4.1: Resultados obtenidos de la implementación del algoritmo 3.5.1.

En la figura 4.1 se muestran las imágenes resultantes de la aplicación del algoritmo; donde las imágenes de los incisos (a),(b) y (c) pertenecen al estacionamiento centro; mientras que los incisos (d), (e) y (f) corresponden al estacionamiento izquierdo; los incisos (h), (g) e (i) al estacionamiento derecho y (j), (k) y (l) al estacionamiento lateral.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.1: Resultados de la implementación del algoritmo 3.5.1.

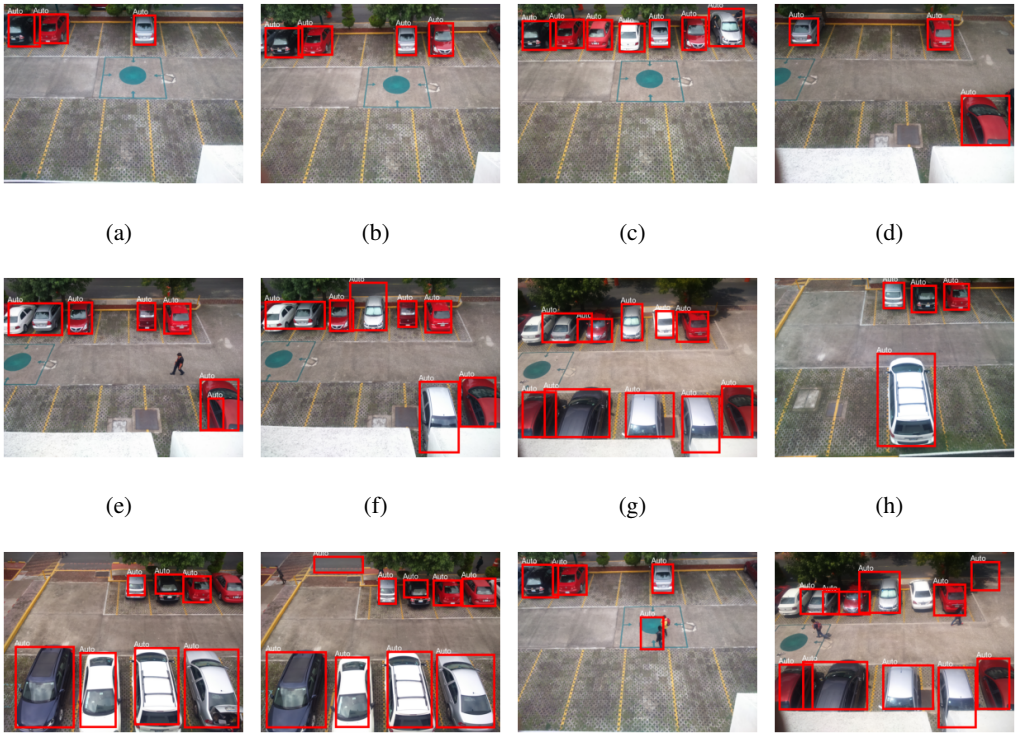
4.2. Resultados de algoritmo 3.5.2

A continuación, se presentan los resultados de la implementación del algoritmo 3.5.2, el cual está basado en el método de segmentación propuesto en la subsección 3.2.2; para la extracción de características, se emplea el método basado en la obtención de la firma ($G[n]$), propuesto en la subsección 3.3.1; y para la etapa de identificación se hace uso del método presentado en la subsección 3.4.1.

Los resultados obtenidos durante la implementación del algoritmo 3.5.2 son presentados a continuación, el cual fue evaluado con 86 imágenes. En la figura 4.2 se muestran algunas capturas de los resultados obtenidos, en donde se encierran en recuadros de color rojo los objetos detectados como automóviles. En la tabla 4.2, se muestran los resultados de la evaluación para las 86 imágenes; donde se tienen un total de 504 automóviles, de los cuales se logró detectar 483; debido a que en la mayoría de las imágenes de prueba se obtuvo el reconocimiento de todos los automóviles, la efectividad obtenida fue de 95.83 %, sin embargo a causa de la presencia de falsos positivos, la precisión obtenida fue del 92 %; y el promedio del tiempo de ejecución del algoritmo por imagen fue de 3.16 segundos. Además, también se obtuvo el porcentaje de detección de los automóviles por imagen, esto es, si una imagen contiene un total de 4 automóviles, y el sistema logra detectar a los cuatro, entonces se dice que se alcanzó un 100 % en la detección, mientras que, con respecto a la misma imagen se detectan unicamente 3 autos, entonces se dice que el sistema alcanzó un 75 % en la detección; utilizando esta métrica, se logró alcanzar el 100 % para imágenes con un ambiente de iluminación homogéneo, o bien con cambios ligeros en aumento o disminución de iluminación; en cambio cuando se presentan efectos de sombra debido a un aumento elevado de iluminación, el porcentaje de detección disminuye hasta alcanzar el 80 %; esto se puede apreciar en la figura 4.2, en donde las imágenes del inciso (k) y (l), presentan una baja detección de los automóviles debido a un aumento en la iluminación, además se hace una detección de falsos positivos, debido a objetos que son formados por las sombras que generan los árboles, y de igual manera, en las imágenes que se presentan en los incisos (i), (j) y (l), se detectan falsos positivos, que corresponden a señalamientos dentro del estacionamiento y parte de la calle que se encuentra frente al mismo.

Imágenes	Autos	Detectados	Efectividad	Precisión	Tiempo
86	504	483	95.83 %	92.7 %	3.16 segs.

Tabla 4.2: Métricas de los resultados obtenidos con las imágenes de prueba.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figura 4.2: Resultados de la implementación del algoritmo 3.5.2.

4.3. Resultados de algoritmo 3.5.3

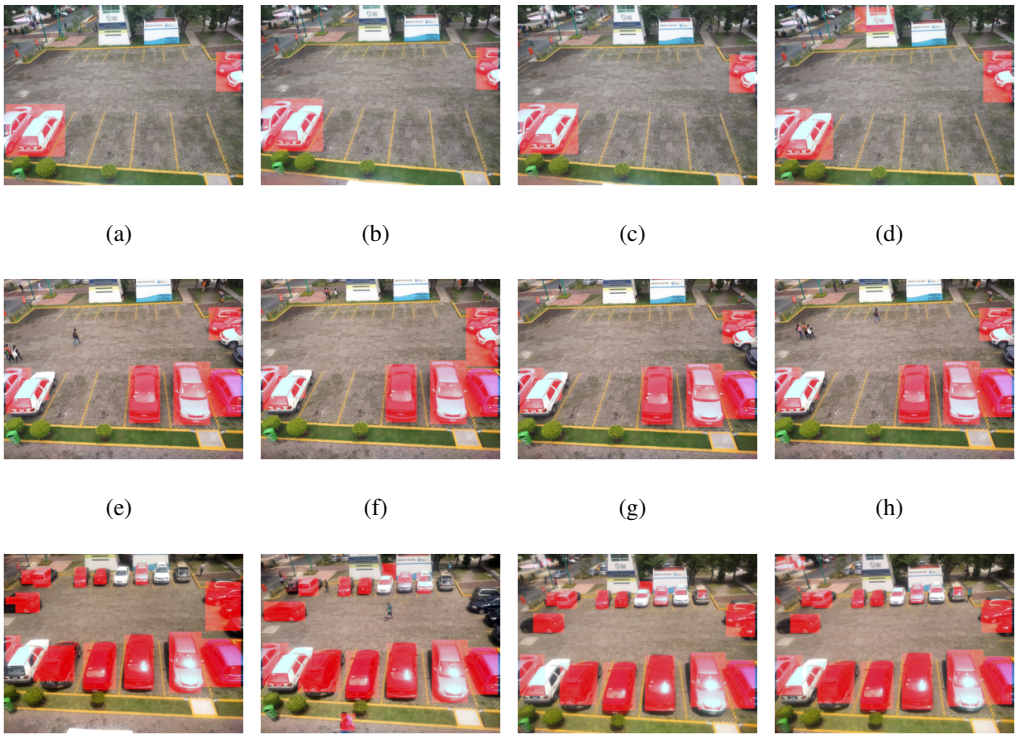
En el algoritmo 3.5.3, se presenta un método para la detección de automóviles para las imágenes de la parte lateral del estacionamiento, ya que éstas no fueron incluidas durante la implementación del algoritmo 3.2.2, debido a que con este tipo de imagen, se obtuvo una baja efectividad a causa de los autos de menor tamaño. El método que se presenta en el algoritmo 3.5.3, está basado en el proceso de segmentación que se muestra en la subsección 3.2.3; para la extracción de características se hace uso del método presentado en la subsección 3.3.2, donde se emplean descriptores de textura mediante la matriz de co-ocurrencia (GLCM), agregando el área y la media de los valores de intensidad del objeto en RGB; y para la etapa de clasificación, se utiliza el método presentado en la subsección 3.4.2, donde se hace uso de la red MLP, el cual se configura a una arquitectura piramidal, con 35 neuronas en la primer capa oculta, 15 en la segunda y una neurona en la capa de salida. Para la evaluación del algoritmo 3.5.3, se utilizaron un total de 57 imágenes de la parte lateral del estacionamiento; en la tabla 4.3, se presentan los resultados obtenidos, donde se consiguió un 90 % de efectividad y 91.4 % en precisión.

En la figura 4.3, se presentan los resultados de detección obtenidos durante la implementación del algoritmo. En los incisos (a), (b), (c) y (d), se muestran los resultados para imágenes donde se

Total de autos	FP	Autos detectados	Efectividad general	Precisión
653	7	594	90 %	91.4 %

Tabla 4.3: Resultados de métricas obtenidos durante la implementación del algoritmo 3.5.3.

tiene una baja cantidad de automóviles en el estacionamiento; en los incisos (e), (f), (g) y (h), se muestran los resultados para imágenes donde se tiene una cantidad moderada de automóviles; y en los incisos (i), (j), (k) y (l), se presentan los resultados para imágenes con alta cantidad de autos.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figura 4.3: Resultados de la implementación del algoritmo 3.5.3.

4.4. Resultados de algoritmo 3.5.4

El método propuesto en el algoritmo 3.5.4, hace uso de la ubicación semiautomática de regiones de interés (ROI) presentado en la subsección 3.2.4, este método se propone como una alternativa, el cual evita utilizar métodos basados en segmentación para la ubicación de los objetos de interés, propuestos en los algoritmos 3.5.1, 3.5.2 y 3.5.3. Las ROI son generadas de

acuerdo con el algoritmo 3.2.4, posteriormente se extraen las características de cada ROI mediante el método presentado en la subsección 3.3.3; y para llevar a cabo la detección, se emplea el método presentado en la subsección 3.4.3, el cual hace uso del clasificador perceptrón multicapa (MLP) con dos capas ocultas, configurado a una arquitectura piramidal, con 11 neuronas en la primer capa oculta y 9 en la segunda, y 2 neuronas en la capa salida, las cuales están conectadas a una función de activación *softmax*.

Para la etapa de evaluación se emplearon 210 imágenes del conjunto presentado en la sección 3.1, tomando en cuenta las cuatro topologías de estacionamiento diferentes (derecha, centro, izquierda y lateral). Se evaluó la velocidad de ejecución, así como, la efectividad (ver ecuación 4.1). En la tabla 4.3, se presentan los resultados de la evaluación del algoritmo.

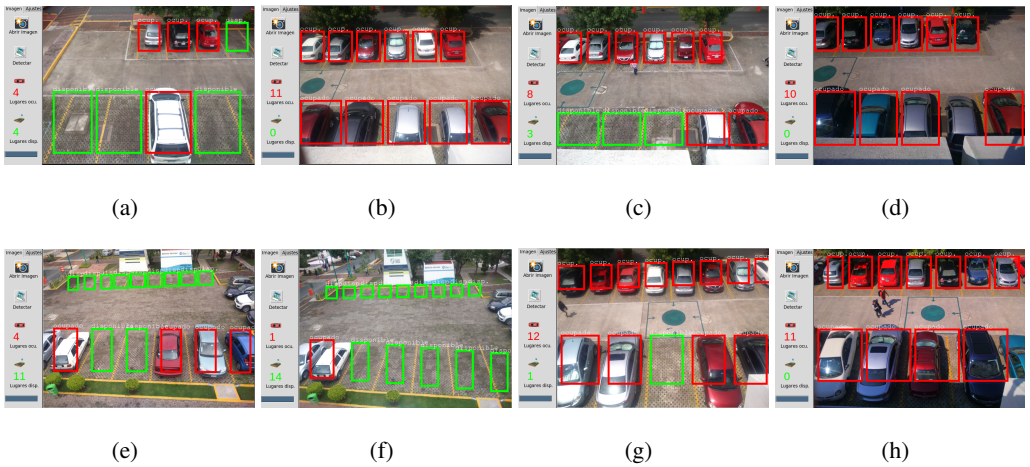
Muestras de evaluación	**Efectividad**	**Precisión**	**Tiempo de ejecución**	**Vel. de CPU**
2009	90 %	92 %	2 segs.	1.5 GHz.

Tabla 4.4: Resultados de métricas obtenidos durante la implementación del algoritmo 3.5.4.

En la figura 4.4, se muestran los resultados de la implementación del algoritmo, donde son enmarcados en color verde los lugares disponibles y en rojo los ocupados. Se logró una adecuada detección de las regiones, esto se puede observar en los incisos (a), (b), (c), (e), (f), (g) y (h); sin embargo, el principal inconveniente observado fue la susceptibilidad a los cambios de posición de la cámara de captura, que puede ocasionar una mala ubicación de las ROI, este caso se puede observar en la imagen del inciso (f), donde la cámara de captura se ha inclinado ligeramente hacia abajo, de tal manera que logra captar el techo del estacionamiento, lo que ocasiona que las ROI de la parte inferior de la imagen combinen la información del techo del estacionamiento y los automóviles, y como resultado de esto, en algunos casos el clasificador no logra las identificar las características de las ROI.

4.5. Resultados de algoritmo 3.5.5

Una vez evaluado los algoritmos 3.5.1, 3.5.2, 3.5.3 y 3.5.4, se observó que la combinación de algunos de estos puede producir un mejor funcionamiento, que permita incluir las cuatro partes del estacionamiento, tal es el caso de los algoritmos 3.5.2 y 3.5.3. De acuerdo con esto, se diseñó el algoritmo 3.5.5, en el cual los valores de los parámetros de la red ICM se establecieron de manera local, es decir, para cada parte del estacionamiento, generando así cuatro diferentes valores de



(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.4: Resultados de la implementación del algoritmo 3.5.4.

los parámetros, donde para la parte centro: $\boldsymbol{f} = 0{,}08$, $\boldsymbol{g} = 0{,}65$ y $\boldsymbol{h} = 20$; para la parte derecha: $\boldsymbol{f} = 0{,}09$, $\boldsymbol{g} = 0{,}68$ y $\boldsymbol{h} = 20$; para la parte izquierda: $\boldsymbol{f} = 0{,}19$, $\boldsymbol{g} = 0{,}61$ y $\boldsymbol{h} = 19$; y para la parte lateral: $\boldsymbol{f} = 0{,}05$, $\boldsymbol{g} = 0{,}68$ y $\boldsymbol{h} = 20$. En la tabla 4.5, se presentan los resultados obtenidos por cada parte del estacionamiento, donde en promedio se alcanzó una efectividad de 95.2 %, la cual es mayor al obtenido por el algoritmo 3.5.3, por otra parte, se obtuvo una precisión promedio de 96.8, que es mayor al conseguido en el algoritmo 3.5.2.

Estacionamiento	Efectividad (%)	Precisión (%)	Total de imágenes
Centro	95.3	99	98
Derecho	94.1	98.8	61
Izquierdo	96.8	99.1	50
Lateral	94.6	91.4	51

Tabla 4.5: Métricas de los resultados obtenidos empleando el algoritmo 3.5.5.

4.6. Resultados del algoritmo general

El algoritmo general se propuso para poder llevar a cabo la detección de espacios disponibles en las cuatro partes del estacionamiento (centro, derecho, izquierdo y lateral), y de igual manera tanto en imágenes con iluminación homogénea, como en imágenes con presencia de sombras. Para esto se generó el algoritmo 3.5.6, el cual integra los métodos de detección de autos presentados previamente; donde de acuerdo al tipo de imagen de entrada, se utiliza el algoritmo 3.5.5 o 3.5.4. En la tabla 4.6, se presentan los resultados obtenidos del algoritmo general, los cuales son el

promedio de los resultados de detección obtenidos por los métodos presentados en los algoritmos 3.5.5 y 3.5.4; donde éste último se evaluó unicamente con las imágenes donde se tiene presencia de sombra, siendo un total de 54, obteniendo 99.2 % de efectividad y 99.4 % en precisión. Además, se cotejan los resultados obtenidos por los algoritmos 3.5.2, 3.5.3 y 3.5.4.

	Algoritmo general 3.5.6	**Algoritmo 3.5.2**	**Algoritmo 3.5.3**	**Algoritmo 3.5.4**
Imágenes de aprendizaje	66	36	6	30
Imágenes de validación	260	87	57	270
Total de imágenes	380	123	63	300
Parte del estacionamiento	centro, derecho, izquierdo y lateral	centro, derecho e izquierdo	lateral	centro, derecho, izquierdo, lateral
Método	Segmentación por ICM	Segmentación por ICM	Segmentación por ICM mediante propagación	ROI
Características	Firma ($G[n]$) en RGB	Firma ($G[n]$) en escala de grises	GLCM	Firma ($G[n]$) en RGB
Efectividad (%)	97.2	95.8	90	90

Tabla 4.6: Métricas de los resultados obtenidos con los algoritmos propuestos.

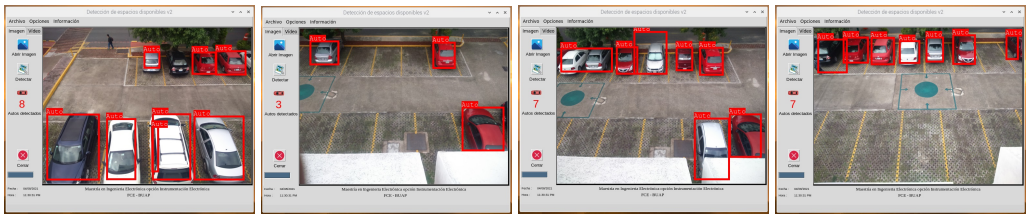
El algoritmo 3.5.4, fue empleado para la detección de espacios disponibles y ocupados en las cuatro partes del estacionamiento, obteniendo 90 % de efectividad (ver tabla 4.6). Este método presentó baja susceptibilidad a condiciones de baja iluminación o efectos de sombra, sin embargo, el principal inconveniente y causante de bajar la efectividad durante las pruebas realizadas, es la susceptibilidad a los movimientos de la cámara de captura, que provoca una mala ubicación de las ROI. El algoritmo 3.5.3 se empleó para la detección de automóviles en la parte lateral del estacionamiento, donde se obtuvo un 90 % de efectividad (ver tabla 4.6); las causas principales de alcanzar dicho nivel de efectividad se deben a: automóviles de menor tamaño y tonalidad similar al piso del estacionamiento en los mismos. Por otro lado, el algoritmo 3.5.2 se generó para la detección de autos en las partes centro, derecha e izquierda del estacionamiento, obteniendo una

efectividad de 95.8 % en imágenes con iluminación homogénea, sin embargo, el algoritmo presenta un bajo desempeño en condiciones de baja iluminación y efectos de sombras.

El algoritmo 3.5.6 obtuvo 97.2 % de efectividad debido a la combinación de los algoritmos 3.5.5 y 3.5.4, siendo que para el primero se obtuvo 95.2 % de efectividad promedio (ver tabla 4.5), y para el segundo 99.2 %.

4.7. Resultados de implementación del algoritmo en Raspberry Pi

En esta sección se presentan los resultados obtenidos de la implementación de la interfaz de usuario en la tarjeta *Raspberry Pi 4*. En la figura 4.5 se presentan algunas capturas de pantalla de la ejecución de la interfaz de usuario.



(a) (b) (c) (d)

Figura 4.5: Resultados de la implementación de la interfaz en la tarjeta *Raspberry Pi* para la detección de automóviles.

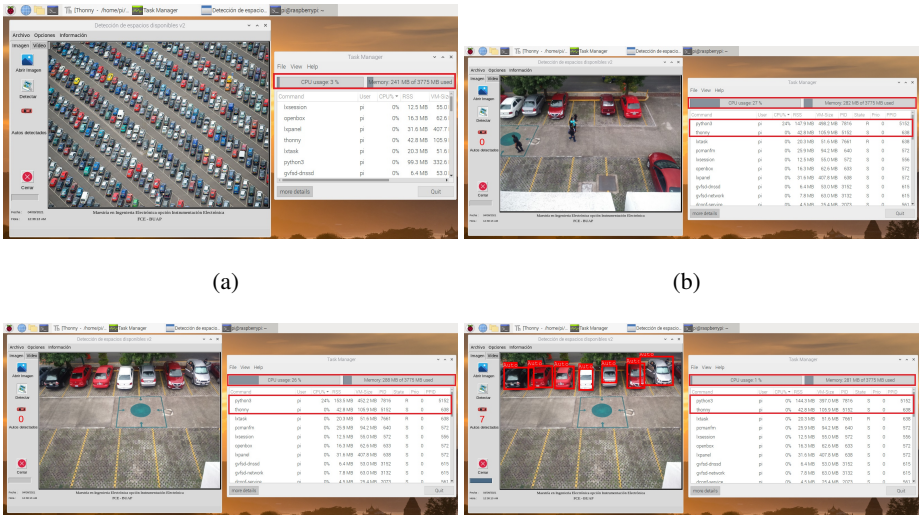
Se evaluó el funcionamiento de la tarjeta en base a la velocidad de ejecución, en la tabla 4.7 se presentan los tiempos de ejecución obtenidos haciendo uso del algoritmo general. Donde se tiene un rango de velocidad de ejecución entre 5.92 y 14.5 segundos, el cual se considera adecuado para su implementación en un estacionamiento, tomando en cuenta el tiempo mencionado. La velocidad de ejecución depende de los recursos de procesamiento en donde esté implementado.

Imágenes	Tiempo promedio	Tiempo máximo	Tiempo mínimo
86	8.23 seg.	14.5 seg.	5.92 seg.

Tabla 4.7: Métricas de los resultados obtenidos con las imágenes de prueba.

Además de evaluar el tiempo de ejecución, también se midió la cantidad de memoria y CPU utilizados al momento de ejecutar el algoritmo, para esto se empleó el administrados de tareas que ofrece el SO *Raspbian*, en la figura 4.6 se muestran capturas del administrador de tareas al

momento de estar ejecutando el algoritmo para la detección de los automóviles y al estar únicamente ejecutando la interfaz de usuario, ésto con el objetivo de observar la cantidad de memoria y CPU empleados; en el inciso (a) se presenta la captura al momento de iniciar la ejecución de la interfaz, donde el porcentaje de CPU utilizado se encuentra oscilando entre 1% y 3%, mientras que la memoria presenta un total de 241 MB; en el inciso (b), se presenta la captura al momento de estar ejecutando el algoritmo general, y se puede apreciar un incremento de los recursos, el uso de CPU alcanza un 27% y la memoria 282 MB, ya que el interprete de *Python 3* demanda un uso del 24% del CPU; en la imagen del inciso (c) se presenta el mismo escenario, donde el interprete *Python 3* demanda un porcentaje del CPU similar, un total de 24%; por otro lado, en la captura del inciso (d) se ha terminado de ejecutar el algoritmo de detección de automóviles, y ahora unicamente se encuentra ejecutando la interfaz de usuario, en este escenario disminuye el uso de CPU a valores similares al presentado en la imagen del inciso (a), pero al contrario de ésta, la cantidad de memoria utilizada permanece a niveles similares a las capturas presentadas en los incisos (b) y (c).



(a) (b)

(c) (d)

Figura 4.6: Capturas de pantalla del funcionamiento de la interfaz de usuario y el algoritmo para detección de automóviles.

Capítulo 5

Conclusiones

De acuerdo con los resultados observados de la implementación de los algoritmos propuestos se presentan las siguientes conclusiones.

El algoritmo de detección de automóviles mediante segmentación por zonas obtuvo un reconocimiento de todos los autos en imágenes con iluminación homogénea. Sin embargo, en imágenes con efectos de sombra u oclusión entre los autos, se observó que el algoritmo puede detectar dos autos como un solo objeto; esto se puede observar en imágenes con presencia de sombras producidas por un incremento de iluminación. Por último, se detectó un bajo reconocimiento de los automóviles, debido a que el área de interés seleccionado se ubicó de forma incorrecta, a causa del movimiento de la cámara de captura.

En el caso del algoritmo de detección de autos mediante $G[n]$ y MLP se presentó un buen desempeño en imágenes con iluminación homogénea, o bien, con ligeros cambios en aumento o disminución de iluminación, alcanzando un 95.8 % de efectividad, sin embargo, en imágenes con presencia de sombras la efectividad es afectada, decayendo hasta un 60 %. Lo anterior es debido a los efectos de sombras, las cuales son generadas a causa de un incremento de iluminación generalmente debido a la luz del sol, donde algunos objetos como los árboles que se encuentran alrededor del estacionamiento y los automóviles mismos, producen su sombra respectiva, dando como resultado patrones de firma que pueden ser detectados como falsos positivos o verdaderos negativos, además, las sombras producidas por los automóviles pueden propiciar a que las estructuras de los mismos se junten provocando así que se detecten dos o más automóviles como uno solo.

El algoritmo de detección de autos mediante GLCM y MLP fue propuesto para llevar a cabo la detección de automóviles en las imágenes de la parte lateral del estacionamiento, ya que en éstas el algoritmo de detección de autos mediante $G[n]$ y MLP, presentó un bajo rendimiento. El método propuesto generó resultados adecuados en la detección de automóviles, donde obtuvo un 90 % de efectividad general y se alcanzó hasta el 100 % de efectividad por imagen. Sin embargo, en algunas imágenes se observó que estructuras externas se pueden detectar como un auto (falso

positivo), debido un ligero incremento de iluminación en la imagen, además, los autos de menor tamaño en cinco imágenes de prueba no pudieron ser detectados, esto debido a dos aspectos, el primero es que en el proceso de segmentación el chasis de los automóviles con tonalidad similar al piso del estacionamiento es removido, y el segundo se debe a la oclusión, que al presentarse en los automóviles de menor tamaño evita extraer características relevantes de los mismos.

De manera alternativa a emplear un método de segmentación para poder así extraer los objetos de interés dentro de una imagen, se diseñó el algoritmo para detección de espacios disponibles mediante ROI, $G[n]$ y PCA, el cual implementa un método basado en la ubicación semiautomática de regiones de interés, para que a partir de éstas, se pueda identificar si un espacio de estacionamiento se encuentra ocupado o no. Este algoritmo toma en cuenta las cuatro partes del estacionamiento, las cuales son: parte izquierda, centro, derecha y lateral. Se obtuvo un 90 % de efectividad, presentando robustez a los cambios de iluminación así como a los efectos de sombra.

El algoritmo general permite llevar a cabo la detección de automóviles en las cuatro partes de estacionamiento, donde para imágenes con iluminación homogénea se obtiene una efectividad del 95.2 % y para imágenes con presencia de sombras se obtuvo 99.2 %. Para imágenes homogéneas se empleó un método basado en la red ICM para segmentación y extracción de características, mientras que para la clasificador se empleó la red MLP. Por otro lado, para imágenes con efectos de sombras se hace uso del método basado en ubicación de ROI, la firma $G[n]$ y MLP.

De lo anteriormente mencionado se concluye que los algoritmos propuestos permiten llevar a cabo la detección correcta de los automóviles, bajo ciertas condiciones, por ejemplo, para el algoritmo de detección de automóviles mediante segmentación por zonas, se presentan buenos resultados en imágenes con iluminación homogénea, sin embargo presenta alta susceptibilidad al movimiento de la cámara de captura. Por otro lado, en el algoritmo de detección de autos mediante $G[n]$ y MLP, se presenta un bajo rendimiento en imágenes con cambios de iluminación y efectos de sombra; un caso similar ocurre con el algoritmo de detección de autos mediante GLCM y MLP, donde los cambios de iluminación permiten que estructuras externas al estacionamiento sean detectadas. El algoritmo para detección de espacios disponibles mediante ROI, $G[n]$ y PCA, presenta susceptibilidad a los cambios de posición de la cámara de captura, ya que al presentarse esto, la ubicación de regiones de interés puede ser errónea. A pesar de los inconvenientes mencionados, se logró llevar a cabo de manera adecuada el objetivo de este trabajo, alcanzando con el algoritmo general, una efectividad promedio de 97.2 %, esto en base a la integración de los algoritmos que combinan la detección de autos mediante $G[n]$ y MLP, y ubicación de ROI, $G[n]$ y PCA, donde cada uno opera con el tipo de imágenes donde obtiene un mejor rendimiento.

Con el objetivo de tener un mejor control en la activación de las neuronas de la red ICM, se llevó a cabo una propuesta de modificación en la obtención de la información de las neuronas

vecinas, siendo que si el resultado de la convolución entre éstas y la salida de una neurona en un tiempo anterior es mayor a cero, entonces a dicha neurona se le aplica un valor de incremento, el cual se ha denominado *conv*. Por otra parte, se llevaron a cabo pruebas de clasificación con RBF, MLP, KNN y wKNN, donde los clasificadores RBF y wKNN presentaron baja efectividad en la clasificación de las características empleadas en este trabajo, debido a esto se descarto el uso de los mismos.

El uso de la tarjeta *Raspberry Pi 4* durante la implementación del algoritmo propuesto para la detección de automóviles en un estacionamiento, presentó resultados favorables que permiten la implementación de este sistema en un estacionamiento, considerando los tiempos empleados para la etapa de procesamiento.

5.1. Trabajo futuro

En base a los resultados observados, se propone como trabajo futuro los siguientes puntos:

- Desarrollar un método de segmentación basado en la red ICM que presente baja susceptibilidad a los cambios de iluminación, mediante la adaptación automática de sus parámetros.
- Recolectar una mayor cantidad de imágenes del estacionamiento las cuales contengan diferentes escenarios, como por ejemplo, imágenes del estacionamiento vacío, autos con diferentes tonalidades de color y condiciones de iluminación que se pueden presentar a lo largo del día.
- Implementar un clasificador en cascada en combinación con la exploración de diferentes descriptores, como el histograma de gradientes orientados (HOG) y las redes neuronales convolucionales (CNN), que permitan mejorar la detección de los objetos.
- Desarrollar un método para la generación de ROI de manera automática.
- Explorar métodos basados en aprendizaje profundo para la detección de automóviles con el conjunto de imágenes del estacionamiento.
- Adaptar el algoritmo propuesto para la detección de motocicletas en el estacionamiento.

Bibliografía

[1] Feature extraction using unit-linking poulse coupled neural network and its applications. *Springer science*, 2007.

[2] Arabic sign language recognition system based on adaptive pulse-coupled neural network. *AMLTA*, 2012.

[3] Leaf recognition based on pcnn. *Natural computing applications forum*, 2015.

[4] Sayani Banerjee, TS Ashwin, and Ram Mohana Reddy Guddeti. Automated parking system in smart campus using computer vision technique. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 931–935. IEEE, 2019.

[5] Nazia Bibi, Muhammad Majid, Hassan Dawood, and Ping Guo. Automatic parking space detection system. 10 2017.

[6] Bill Yang Cai, Ricardo Alvarez, Michelle Sit, Fábio Duarte, and Carlo Ratti. Deep learning-based video system for accurate and real-time parking measurement. *IEEE Internet of Things Journal*, 6(5):7693–7701, 2019.

[7] Erik Cuevas, Daniel Zaldívar, and Marco Pérez. *Procesamiento digital de imágenes con MATLAB & Simulink*. Ra-Ma, 2016.

[8] Ivan Nunes da Silva. *Artificial neural networks*. Springer, [Cham], 2017.

[9] E Roy Davies. *Computer and machine vision: theory, algorithms, practicalities*. Academic Press, 2012.

[10] Richard Edmondson, Michael Rodgers, and Michele Banish. Using a genetic algorithm to find an optimized pulse coupled neural network solution, 2008.

[11] Ulf Ekblad and Jason Kinser. Theoretical foundation of the intersecting cortical model and its use for change detection of aircraft, cars, and nuclear explosion tests. *Signal Processing*, 84:1131–1146, 07 2004.

[12] Rosenblatt F. The eperceptron: Ea eprobabilistic emodel for einformation estorage and eorganization in the brain, 1958.

[13] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.

[14] Juanita Hernández and Wilfrido Gómez. Automatic tuning of the pulse-coupled neural network using differential evolution for image segmentation, 2016.

[15] C. Huang, Y. Tai, and S. Wang. Vacant parking space detection based on plane-based bayesian hierarchical framework. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(9):1598–1610, September 2013.

[16] H. G. Jung. Semi-automatic parking slot marking recognition for intelligent parking assist systems. *The Journal of Engineering*, 2014(1):8–15, 2014.

[17] Soomok Lee and Seung-Woo Seo. Available parking slot recognition based on slot context analysis. *IET Intelligent Transport Systems*, 10(9):594–604, 2016.

[18] Haiyan Li, Lei Guo, Pengfei Yu, Jianhua Chen, and Yiying Tang. Image segmentation based on iterative self-organizing data clustering threshold of pcnn, 2016.

[19] Thomas Lindblad, Jason M Kinser, and JG Taylor. *Image processing using pulse-coupled neural networks*. Springer, 2013.

[20] Yide Ma, Kun Zhan, and Zhaobin Wang. *Applications of pulse coupled neural networks*. Springer, Berlin [u.a.], 2010. Literaturangaben.

[21] Stephen Marsland. *Machine learning*. A Chapman and Hall book. CRC Press/Taylor and Francis, Boca Raton, Fla., second edition edition, 2015.

[22] Chin-Kit Ng, Soon-Nyean Cheong, and Yee-Loo Foo. Lightweight deep neural network approach for parking violation detection. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, pages 332–337, 2018.

[23] Nobuyuki Otsu. A threshold selection method from gray-level histograms. 9:62–66, 1979.

[24] Elena Polycarpou, Lambros Lambrinos, and Eftychios Protopapadakis. Smart parking solutions for urban areas. In *2013 IEEE 14th International Symposium on.ª World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2013.

[25] G Revathi and VR Sarma Dhulipala. Smart parking systems and sensors: A survey. In *2012 International Conference on Computing, Communication and Applications*, pages 1–5. IEEE, 2012.

[26] Simon Rogers and Mark Girolami. *A first course in machine learning*. A Chapman and Hall book. CRC Press, Boca Raton, Fla. [u.a.], 2011.

[27] David E. Rumelhart. *Parallel distributed processing, 9th Edition*. MIT Pr., 1989.

[28] Kevin Aguilar Dominguez Manuel Mejia Lavalle Humberso Sossa. Mejora eficiente de la luminosidas en imágenes del cerebro humano utilizando redes neuronales de pulso-acoplado. *Computación y Sistemas*, 24:105,120, 2019.

[29] Jae Kyu Suhr and Ho Gi Jung. Fully-automatic recognition of various parking slot markings in around view monitor (avm) image sequences. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 1294–1299. IEEE, 2012.

[30] Paula Tatulea, Florina Calin, Remus Brad, Lucian Brâncovean, and Mircea Greavu. An image feature-based method for parking lot occupancy. *Future Internet*, 11(8):169, 2019.

[31] Luo-Wei Tsai, Jun-Wei Hsieh, and Kuo-Chin Fan. Vehicle detection using normalized color and edge map. *IEEE transactions on Image Processing*, 16(3):850–864, 2007.

[32] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[33] Seema Wazarkar, Bettahally N. Keshavamurthy, and Ahsan Hussain. Region-based segmentation of social images using soft knn algorithm. 125:93–98, 2018.

[34] Shi Weili, Miao Yu, Chen Zhanfang, and Zhang Hongbiao. Research of automatic medical image segmentation algorithm based on tsallis entropy and improved pcnn, 2009.

Apéndice A
Publicaciones

A.1. Participación en el Congreso Internacional de Innovación Tecnológica y Computación (CIITEC 2020)

Se participó con el artículo titulado: ***Segmentación de lugares disponibles en un estacionamiento haciendo uso de redes neuronales pulso-acopladas***.

Actualmente se encuentra publicado en Pre-print a la espera de asignación de páginas.

Datos de la publicación:

- Nombre de la revista: Pistas educativas (indexada en Latinindex).
- ISSN: 2448848X.
- Número de la revista 139.
- Volumen 43.
- Año de publicación: 2021.
- Autores: Víctor Romero Bautista, Aldrin Barreto Flores, Salvador E. Ayala Raggi, José F. Portillo Robledo y Verónica E. Bautista Lopéz.

A.2. Participación en el 8vo. Congreso Internacional de Robótica y Computación (CIRC 2021)

Se participó con el trabajo titulado: ***Detección de automóviles en un estacionamiento haciendo uso de las redes neuronales pulsantes***.

Datos de la publicación:

- El artículo se encuentra publicado en el libro del congreso.
- ISBN del libro: 978-607-98174-9-7
- Páginas en las que se encuentra artículo: de 87 a 95.
- Año de publicación: 2021.
- Autores: Víctor Romero Bautista, Aldrin Barreto Flores, Salvador E. Ayala Raggi, José F. Portillo Robledo y Verónica E. Bautista Lopéz.

A.3. Participación en la XVII Semana Nacional de Ingeniería Electrónica y II Semana Iberoamericana de Ingeniería Electrónica (SENIE 2021)

Se participó con el trabajo titulado: ***Detección de automóviles en un estacionamiento utilizando red ICM y GLCM***.

Datos de la publicación:

- Se presentó el trabajo durante el evento el cual se llevó a cabo del 17 al 19 de Noviembre del 2021.
- La publicación del artículo se llevará a cabo en la revista Pista Educativas (indexada en Latinindex) durante la última semana de diciembre del presente año.
- Autores: Víctor Romero Bautista, Aldrin Barreto Flores, Salvador E. Ayala Raggi, Verónica E. Bautista Lopéz y José F. Portillo Robledo.

A.4. Participación en el III Congreso Internacional y XI Congreso Nacional de Ciencias de la Computación (CONACIC 2021)

Se participó con el trabajo titulado: ***Detección de lugares disponibles en un estacionamiento***.

Datos de la publicación

- Se llevó a cabo la presentación del trabajo el 27 de octubre del 2021.
- La publicación del artículo se llevará a cabo en un libro con ISBN.
- Autores: Víctor Romero Bautista, Aldrin Barreto Flores, Salvador E. Ayala Raggi y Verónica E. Bautista Lopéz.