Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Postgrado en Ciencias Matemáticas

Reconocimiento de placas basado en Modelos Ocultos de Markov

Tesis

Presentada para obtener el grado de
Maestro en Ciencias Matemáticas

Presenta
Gustavo Portillo Ramírez

Director de Tesis
Dr. Hugo Adán Cruz Suárez

Puebla, Puebla. Abril 2021

DR. SEVERINO MUÑOZ AGUIRRE
SECRETARIO DE INVESTIGACIÓN Y
ESTUDIOS DE POSGRADO, FCFM-BUAP
P R E S E N T E:

Por este medio le informo que el C:

GUSTAVO PORTILLO RAMÍREZ

estudiante de la Maestría en Ciencias (Matemáticas), ha cumplido con las indicaciones que el Jurado le señaló en el Coloquio que se realizó el día 26 de marzo de 2021, con la tesis titulada:

Reconocimiento de placas basado en Modelos Ocultos de Markov

Por lo que se le autoriza a proceder con los trámites y realizar el examen de grado en la fecha que se le asigne.

A T E N T A M E N T E.
H. Puebla de Z. a 12 de abril de 2021

DRA. PATRICIA DOMÍNGUEZ SOTO
COORDINADORA DEL POSGRADO
EN MATEMÁTICAS.

Facultad
de Ciencias
Físico Matemáticas

Av. San Claudio y 18 Sur, edif. FM1
Ciudad Universitaria, Col. San
Manuel, Puebla, Pue. C.P. 72570
01 (222) 229 55 00 Ext. 7550 y 7552

Dedicado a
mi familia

Agradecimientos

A mis padres: Teresa y Jacinto, a mis hermanas: Amada, Bertha y Reyna, a mis sobrinas, a mis hermanos: Edmundo, Francisco, Gilberto y Josué, por darme ánimo, compañía y motivación.

A mi novia Sinai por sus consejos, amor, apoyo, ánimo, cariño, por sus chistes y por estar en mi vida.

A mi asesor de tesis, Dr. Hugo Adán Cruz Suárez, a quien agradezco totalmente su apoyo, tiempo, paciencia y por los conocimientos compartidos en sus cursos, así como la atención prestada al trabajo de tesis.

A mis sinodales: Dra. Hortensia Josefina Reyes Cervantes, Dr. Víctor Hugo Vázquez Guevara, Dr. Francisco Solano Tajonar Sanabria y Dra. María del Rocio Ilhuicatzi Roldán por su tiempo para revisar este trabajo, por sus sugerencias y observaciones que ayudaron a mejorarlo.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca otorgada durante estos años para la obtención de grado.

A mis compañeros y amigos: América, Cristhian, Edgar, Fernando, Jaicer, Julio, Roque.

Introducción

El desarrollo de sistemas de reconocimiento de imágenes ha permitido que las máquinas lleven a cabo tareas que consideramos sencillas, pero que en la actualidad son de suma importancia, por ejemplo, ¿cómo sería la vida sin un detector de código de barras o sin un lector de huellas dactilares? La base del reconocimiento de imágenes es el reconocimiento de patrones de imágenes, donde pensamos en un patrón como una disposición espacial de características. Una clase de patrón se define como un conjunto de patrones que tienen propiedades en común, de esta manera, dado un conjunto de patrones cuya clase se desconoce, el trabajo de un sistema de reconocimiento de patrones es asignar una etiqueta de clase a cada uno de los patrones de entrada. En el campo de las aplicaciones de transporte inteligente, la detección y reconocimiento automático de placas a partir de imágenes fijas o secuencias de videos es un tema destacado de la tecnología de visión por computadora y reconocimiento de patrones ya que es una tarea importante en el transporte y la vigilancia que tiene usos prácticos y relevantes como lo es aplicar la ley de tránsito, detección de vehículos robados, control de flujo de tránsito, etc.

Debido a las variantes en las condiciones de la adquisición de la imagen y a que los diseños de las placas varían de un país a otro, la detección de la placa sigue siendo un problema abierto. Por este motivo, se aborda el problema de detección y reconocimiento de placas para vehículos con placa del estado de Puebla. En la actualidad, se han realizado muchos esfuerzos por obtener un reconocimiento automático ideal de los caracteres de la placa de un auto. Existen varios enfoques para enfrentar el problema de reconocimiento como la comparación de plantillas [14], entrena-

miento de redes neuronales [2] y redes neuronales multicapa [20]. Sin embargo, antes del reconocimiento, surge un problema: dada la imagen ¿cómo obtenemos la región de la imagen que contiene la placa? Algunos investigadores resuelven este problema por medio de la conversión a escala de grises de la imagen, seguido de una aplicación de filtros que resaltan los bordes de la imagen y aprovechan el color de la placa para rescatarla [26]. Ya preprocesada la imagen, algunos autores binarizan la imagen y aplican operaciones morfológicas para reducir el ruido presente y obtienen una región candidata para la placa [3], mientras otros aplican la transformación de Hough [23]. Recientemente, se ha incorporado el uso de redes neuronales convolucionales para enfrentar la localización de la placa [19], [20]. Al indagar en las propuestas bibliográficas para resolver este problema, se encontraron algunos inconvenientes, debido a que no podemos explotar las características de color ya que las placas del estado de Puebla son blancas y se pueden confundir fácilmente con otra región de la imagen. Por otra parte, al aplicar el enfoque de morfología matemática, varias literaturas [12], [21], [24], carecen de las dimensiones de los elementos estructurantes utilizados, de modo que la ubicación parece casi intratable. Al aprovechar el enfoque de la transformación de Hough el costo computacional es excesivo, además, como resultado se obtienen regiones candidatas que tienen formas rectangulares que pudieran no corresponder a la placa.

En este trabajo, para ubicar la región de la placa se propone el Algoritmo de Reducción Suave (ARS), el cual está basado en dos funciones que suavizan la información de las imágenes, después en los datos resultantes se explota el hecho de que los segmentos verticales son más largos en los caracteres de la placa. Posteriormente, se aplica el esqueleto morfológico a la región obtenida de la placa y se procede a realizar una segmentación que consiste en descomponer la imagen en sub-regiones ajenas, en este caso, se busca que cada sub-región contenga un carácter. Finalmente, procedemos al reconocimiento de caracteres a través de los Modelos Ocultos de Markov (MOMs) [5], [7], [8], [15], donde buscamos maximizar la probabilidad del vector de observación de la secuencia de caracteres que constituyen la placa. En este trabajo proponemos un algoritmo, basado en Modelos Ocultos de Markov que será iden-

tificado como algoritmo ARM, que cubre el reconocimiento de los caracteres con MOMs. El reconocimiento a través de los MOMs fue motivado por [17], la ventaja de nuestra propuesta es que fue realizado en un software libre. En la etapa de entrenamiento del reconocimiento, cada una de las imágenes de la placa se modela mediante la estimación de los parámetros de MOM para un conjunto dado de observaciones. El vector de características se obtiene en base a la región de la placa, donde se extraen 4 características de ángulo central por cada carácter.

Índice general

Agradecimientos **III**

Introducción **V**

1. Formato de imagen **1**
1.1. Encabezado de archivo 1
1.2. Encabezado de imagen 2
1.3. Paleta de colores 3
1.4. Datos de los píxeles 5

2. Conceptos teóricos para el reconocimiento **7**
2.1. Preprocesamiento de la imagen 7
2.2. Segmentación de imágenes 8
2.2.1. Detección de bordes 9
2.2.2. Umbralización 12
2.2.3. Filtrado Gaussiano 13
2.3. Morfología Matemática 13
2.3.1. Erosión y dilatación 14
2.3.2. Apertura y cerradura 15
2.3.3. Esqueleto morfológico 17
2.4. Modelos Ocultos de Markov 19
2.4.1. Los tres problemas básicos para MOMs . . 20
2.4.2. Solución al primer problema 21
2.4.3. Solución al segundo problema 24
2.4.4. Solución al tercer problema 26

3. Etapas del reconocimiento **29**
3.1. Preprocesamiento y segmentación de la imagen de entrada 29

3.2. Ubicación de la placa 33
3.3. Reconocimiento de los caracteres con MOMs . . . 38
3.4. Resultados 54

4. Conclusiones **57**

Bibliografía **57**

Capítulo 1

Formato de imagen

Los algoritmos propuestos en este trabajo de tesis reciben la imagen que contiene un auto en formato BMP, por lo tanto, se inicia con el estudio de este formato. En dispositivos digitales, entre ellos, celulares, cámaras y equipos de cómputo se observa una imagen como una representación rectangular que se puede relacionar a una matriz de datos, donde cada entrada de la matriz representa un píxel. Sin embargo, surgen las siguientes preguntas: ¿dónde está la información que determina las dimensiones de la imagen? y ¿dónde queda guardada la información de cuántos bits tiene cada píxel? Las respuestas a estas preguntas las encontramos al indagar en la estructura de la imagen como un archivo.
Al inicio de un archivo de imagen se encuentran especificadas todas las propiedades con que se cuenta: se proporciona el formato, las dimensiones, la cantidad de bits por píxel y lo más importante para el trabajo, a partir de que byte se encuentra la información de los píxeles. A esta parte de la imagen se conoce como encabezado. Los archivos BMP tienen una estructura como la siguiente [4]: encabezado de archivo, encabezado de imagen, tabla de colores y píxeles de datos.

1.1. Encabezado de archivo

En la práctica, los datos de las imágenes primero deben cargarse desde un archivo. Los archivos proporcionan el mecanismo esencial

para almacenar archivar e intercambiar datos de imágenes. Existe una amplia gama de formatos de archivo estandarizados, pero en este trabajo de tesis sólo se considera el formato BMP.
Los valores como el tamaño de la imagen y la codificación de los píxeles suelen estar presentes en el encabezado o cabecera del archivo para facilitar la asignación correcta de memoria.
Todo archivo formato BMP comienza con una estructura BITMAPFILEHEADER cuyo diseño se encuentra en la Tabla 1.1. La función principal de esta estructura es servir como la firma que identifica este formato de archivo. Por otra parte, se pueden verificar 3 puntos para asegurarse que se tenga un archivo BMP:

- Los primeros dos bytes del archivo deben contener los caracteres ASCII “B” y “M”.
- Se puede comparar el tamaño del archivo con el valor en el campo bfSize.
- Los campos bfReserved1 y bfReserved2 deben ser cero.

Campo	Bytes	Descripción
biType	2	Contiene los caracteres “BM”
bfSize	4	Tamaño de archivo
bfReserved1	2	Sin usar
bfReserved2	2	Sin usar
bf0ffBits	4	Desplazamiento al inicio de los datos de píxeles

Tabla 1.1: BITMAPFILEHEADER.

Al utilizar archivos en formato BMP se debe usar el campo bfOffBits para saber cuántos bytes se debe desplazar desde el inicio del archivo para encontrarse con los datos de los píxeles.

1.2. Encabezado de imagen

Lo que sigue es el encabezado o cabecera de imagen y viene en dos formatos distintos, definidos por las estructuras BITMAPINFOHEADER y BITMAPCOREHEADER.

BITMAPCOREHEADER representa el formato BMP de OS/2 y BITMAPINFOHEADER es el formato de Windows mucho más común. La única manera que se tiene para determinar el tipo de estructura de imagen utilizada en un archivo particular es examinar el campo de tamaño de la estructura, que son los primeros 4 bytes de ambos tipos de estructura.
El tamaño de la estructura BITMAPCOREHEADER es de 12 bytes; el tamaño de BITMAPINFOHEADER es al menos de 40 bytes. El diseño de BITMAPINFOHEADER se muestra en la Tabla 1.2. Esta estructura proporciona las dimensiones y la profundidad de bits de la imagen e indica si la imagen está comprimida. La altura de la imagen es un valor sin signo. Un valor negativo para el campo biHeight especifica que los datos de píxeles se ordenan de arriba hacia abajo en lugar de la normal que es de abajo hacia arriba. Las imágenes con un valor negativo de biHeight no se pueden comprimir.

De este modo, se ubican los bytes que pertenecen al biSize, se leen y se avanza el número de bytes guardados para entrar a los datos de los píxeles que proporciona la imagen.
La estructura BITMAPCOREHEADER es el otro formato de encabezado de imagen, su diseño se observa en la Tabla 1.3 y notamos que tiene menos campos y que todos tienen sus análogos a la estructura BITMAPINFOHEADER.

1.3. Paleta de colores

Después del encabezado de archivo lo que sigue es la paleta de colores y puede estar en alguno de tres formatos. Los primeros dos formatos se usan para asignar datos de píxeles a valores de color *RGB* cuando el recuento de bits es 1, 4 u 8 (campos biBitCount o bcCoun). Para archivos BMP en formato Windows, la paleta consta de una matriz de estructuras RGBQUAD de 2 bits, como se observa en la Tabla 1.4. Mientras que los archivos BMP en formato OS/2 utilizan una matriz de estructuras RGBTRIPLE, Tabla 1.5.

El formato final de la paleta de colores no es como tal un mapeo de colores. Si el recuento de bits es 16 o 32 y el valor en el campo biCompression de la estructura BITMAPINFOHEADER es BI-

Campo	Bytes	Descripción
biSize	4	Tamaño de cabecera: al menos 40
biWidth	4	Ancho de la imagen
biHeight	4	Altura de imagen
biPlanes	2	Debe ser 1
biBitCount	2	Bits por píxel: 1, 4, 8, 16, 24 o 32
biCompression	4	Tipo de compresión: $BI - RGB = 0$, $BI - RLE8 = 1$, , $BI - RLE4 = 2$, o $BI - BITFIELDS = 3$
biSizeImage	4	Tamaño de imagen: 0 si no está comprimido
biXPelsPerMeter	4	Resolución preferida en píxeles por metro
biYPelsPerMeter	4	Resolución preferida en píxeles por metro
biClrUsed	4	Número de entradas en el mapa de colores que son realmente usados
biClrImportant	4	Número de colores significativos

Tabla 1.2: Diseño de BITMAPINFOHEADER.

Campo	Bytes	Descripción
bcSize	4	Tamaño de cabecera: debe ser 12
bcWidth	2	Ancho de la imagen
bcHeight	2	Altura de imagen
bcPlanes	2	Debe ser 1
bcBitCount	2	Bits por píxel: 1, 4, 8, o 24

Tabla 1.3: Diseño de BITMAPINFOHEADER.

BITFIELDS, en lugar de una matriz de estructuras RGBQUAD hay una matriz de tres enteros de 4 bytes. Estos tres valores son máscaras de bits que especifican los bits utilizados para los com-

Campo	Bytes	Descripción
rgbBlue	1	Valor de color azul
rgbGreen	1	Valor de color verde
rgbRed	1	Valor de color rojo
rgbReserved	1	Debe ser 0

Tabla 1.4: Estructura RGBQUAD.

Campo	Bytes	Descripción
rgbBlue	1	Valor de color azul
rgbGreen	1	Valor de color verde
rgbRed	1	Valor de color rojo

Tabla 1.5: Estructura RGBTRIPLE.

ponentes rojo, verde y azul. En cada máscara, los bits distintos de cero deben ser contiguos. Además, ningún bit establecido en una máscara de bits puede establecerse en ninguna otra máscara de bits.
Por otra parte las imágenes que utilizan 24 bits, así como las imágenes de 16 y 32 bits sin el campo biCompression establecido en el campo BI-BITFIELDS no tienen una paleta de colores.

1.4. Datos de los píxeles

Se debe utilizar el campo bfOffBits en el encabezado BITMAPFILE para determinar el desplazamiento de la estructura BITMAPFILEHEADER a los datos de los píxeles. Las filas de los píxeles están ordenadas en el archivo de abajo hacia arriba. El número de filas de datos está dado por el campo biHeight o bcHeight en el encabezado de la imagen, mientras que el tamaño de la fila se determina a partir de los campos biBitCount y biWidth o bcBitCount y bcWidth.
En nuestro caso, una imagen de 24 bits por píxel, cada píxel está representado por tres bytes consecutivos que especifican los valores de las componentes azul, verde y rojo, respectivamente.

Una vez identificadas las partes estructurales de la imagen, se puede considerar que la imagen es una matriz bidimensional de píxeles, donde cada píxel consta de un determinado número de bits. La representación más común de imágenes en color es la representación RGB (rojo, verde y azul) la cual se basa en la teoría tricromática de que la sensación de color se produce excitando selectivamente tres clases de receptores en el ojo [1]. El formato que utilizaremos es el formato de color de 24 bits, dejando cada color representado por 8 bits. Por lo tanto, un píxel (x, y) en la imagen a color de 24 bits está representado por $(R_{(x,y)}, G_{(x,y)}, B_{(x,y)})$ donde $R_{(x,y)}, G_{(x,y)}, B_{(x,y)} \in \{0, 1, ..., 255\}$.

Capítulo 2

Conceptos teóricos para el reconocimiento

En este capítulo, se proporcionan los aspectos teóricos en los que están basados los algoritmos propuestos (ARS) y (ARM). Como se ha descrito en el Capítulo 1, se tiene acceso a la información contenida en los píxeles de una imagen dada en formato BMP, lo que sigue es procesar la información, para inicialmente identificar la región de la placa y posteriormente aplicar el reconocimiento de los caracteres. Por tal razón, se dedica la Sección 2.1 al preprocesamiento de imágenes, donde se describen algunos métodos para iniciar con el estudio de los datos de la imagen como el cambio a escala de grises. En la Sección 2.2 se revisan algunos temas relacionados con las segmentación de imágenes como la detección de bordes, la umbralización y un filtro Gaussiano. En la Sección 2.3, se estudian operaciones morfológicas con la finalidad de obtener el esqueleto morfológico de los caracteres. Finalmente, en la Sección 2.4 se revisa la teoría de los MOMs.

2.1. Preprocesamiento de la imagen

Considere una imagen descrita en el sistema RGB, donde la intensidad de cada color está representada por 8 bits, es decir, $R, G, B \in \{0, 1, ..., 255\}$, de este modo, la representación de cada píxel es de 24 bits. El objetivo de esta sección es convertir la imagen de entrada en una imagen en escala de grises, esto ocurre cuando

los valores R, G y B coinciden, es decir, cuando $R = G = B$. El proceso de escala de grises consiste en reducir los 24 bits a solo 8 bits y este se realiza a través de ponderación de RGB o mediante la expresión:

$$f(x, y) = 0.299R_{(x,y)} + 0.587G_{(x,y)} + 0.114B_{(x,y)}, \tag{2.1}$$

donde (x, y) es el píxel con sus correspondientes valores $R_{(x,y)}$, $G_{(x,y)}$, $B_{(x,y)}$ y $f(x, y)$ es el valor gris de este píxel después de la conversión.

2.2. Segmentación de imágenes

Iniciamos con algunos términos requeridos para proporcionar la definición de segmentación [13].

Sea S un subconjunto de píxeles en una imagen. Dos píxeles p y q son llamados conectados en S si existe una trayectoria entre ellos que consiste totalmente de píxeles en S.

Para cualquier píxel p en S, el conjunto de píxeles que están conectados en S es llamado componente conectada de S. Si éste tiene una única componente y la ésta es conectada, entonces S es llamado conjunto conectado.

Sea R un subconjunto de píxeles de una imagen. R es llamada una región de la imagen si es un conjunto conectado. Dos regiones R_i y R_j son adyacentes si su unión forma un conjunto conectado. Si el conjunto formado por la unión de dos regiones no está conectado, la región será llamada disjunta.

Sea R que representa toda la región espacial ocupada por una imagen. Se puede ver la segmentación de imágenes como un proceso que particiona a R en n subregiones, $R_1, R_2, ..., R_n$, tales que

1 $\bigcup_{i=1}^{n} R_i = R$.

2 R_i es un conjunto conectado, para $i = 1, 2, ..., n$.

3 $R_i \cap R_j = \emptyset$ para todo $i, j = 1, 2, ..., n$ con $i \neq j$.

4 $Q(R_i)$ = VERDADERO para $i = 1, 2, ..., n$.

5 $Q(R_i \cup Rj)$ = FALSO para cualesquiera regiones adyacentes R_i y R_j $i, j = 1, 2, ..., n$,

donde $Q(R_k)$ es un predicado lógico definido sobre puntos en el conjunto R_k.
La condición 1 nos dice que cada píxel debe estar en una región. La condición 2 requiere que los puntos en una región estén conectados en algún sentido predefinido. La condición 3 nos dice que las regiones deben ser disjuntas. La condición 4 trata de las propiedades que deben satisfacer los píxeles en una región segmentada, por ejemplo, $Q(R_i) = VERDADERO$ si todos los píxeles de R_i tienen la misma intensidad. Finalmente, la condición 5 indica que dos regiones adyacentes R_i y R_j deben ser diferentes en el sentido del predicado Q.

Por lo tanto, el problema fundamental de la segmentación es dividir una imagen en regiones que satisfagan las condiciones anteriores.
En general, los algoritmos de segmentación de imágenes monocronas se basan en una de dos categorías básicas que se ocupan de las propiedades de los valores de intensidad: discontinuidad y similitud. En la primera categoría se asumen que los límites de las regiones son lo suficientemente distintas entre sí y diferenciadas del fondo de la imagen, para permitir la detección de límites basada en discontinuidades locales en intensidad. Los enfoques de segmentación basados en regiones de similitud se basan en dividir una imagen en regiones que son similares de acuerdo con un conjunto de criterios predefinidos.
Para el caso de estudio nos interesa la segmentación de una imagen en escala de grises, por lo que se aborda la detección de bordes, umbralización y un filtro Gaussiano en las siguientes subsecciones.

2.2.1. Detección de bordes

Generalmente, los objetos y fondo de una imagen tienen diferentes intensidades, por lo tanto, los bordes de objetos son las áreas en que se producen cambios abruptos de intensidad. Un enfoque que se utiliza a menudo para segmentar imágenes en función

de estos cambios abruptos de intensidad (suelen ser locales) es la detección de bordes. Los bordes de una imagen son detectados usando las derivadas de primer y segundo orden. El gradiente de una función imagen $f(x, y)$ es definida como el vector:

$$\nabla f(x,y) = [G_x(x,y), G_y(x,y)]^\intercal = \left[\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)\right]^\intercal.$$

La magnitud de este vector es

$$|\nabla f(x,y)| = \left[\left(\frac{\partial f}{\partial x}(x,y)\right)^2 + \left(\frac{\partial f}{\partial y}(x,y)\right)^2\right]^{\frac{1}{2}}.$$

La propiedad fundamental del vector gradiente es que apunta en la dirección de tasa máxima de cambio del vector imagen f en las coordenadas (x, y). El ángulo en que ocurre esta máxima tasa de cambio es

$$\alpha(x,y) = tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right).$$

En el algoritmo de detección de borde, la suposición es que los bordes corresponden a los píxeles con gradiente alto. Una tasa de cambio de intensidad rápida en la dirección dada por el ángulo del vector gradiente es observado en los bordes de los píxeles.
Considere una imagen en escala de grises $h(x, y)$ y una vecindad de la imagen como:

$$\begin{bmatrix}(x-1,y-1) & (x-1,y) & (x-1,y+1)\\ (x,y-1) & (x,y) & (x,y+1)\\ (x+1,y-1) & (x+1,y) & (x+1,y+1)\end{bmatrix},$$

donde el píxel central es (x, y). Para la detección de bordes se tienen contemplados algunos operadores que están basados en el gradiente de la imagen en escala de grises $h(x, y)$.
El operador tradicional Sobel [13] y [25] es un operador de detección de bordes basados en la longitud del gradiente evaluado en el píxel correspondiente.

Las aproximaciones digitales más simples para las derivadas

parciales [13] usando una máscara (kernel) 3×3 están dadas por

$$\begin{aligned} h_x(x,y) &= h(x+1,y-1) + 2h(x+1,y) + h(x+1,y+1) \\ &\quad - h(x-1,y-1) - 2h(x-1,y) - h(x-1,y+1), \\ h_y(x,y) &= h(x-1,y+1) + 2h(x,y+1) + h(x+1,y+1) \\ &\quad - h(x-1,y-1) - 2h(x,y-1) - h(x+1,y-1). \end{aligned}$$

En esta formulación, la diferencia entre la tercera y la primera fila de la región 3×3 aproxima la derivada en la dirección de x y la diferencia entre la tercera y la primera columna aproxima la derivada en la dirección y.

El operador Sobel, tiene kernel de detección de bordes 3×3 en al menos dos direcciones, que detectan los bordes horizontal y vertical, respectivamente, estas máscaras son

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Cuando se usa el operador Sobel para detectar el borde de una imagen, los kernels de operador horizontal y vertical se usan respectivamente para realizar la operación de convolución en cada píxel de la imagen.
Existen varios tipos de algoritmos que están disponibles para la detección de bordes como detector de bordes Sobel, Prewitt, Robers, Laplaciano de un Gaussiano, Canny [16]. Para el estudio, se usa el detector de bordes Sobel, ya que además de la detección de bordes se agrega una característica de suavizado debido al número dos que aparece en los kernels.

Además, para resaltar bordes más específicos, se puede combinar un kernel con un umbral $s \in \{0, 1, ..., 255\}$ (Subsección 2.2.2), a partir de características que presente la imagen. De este modo, la imagen resultante de la detección de borde es $f(x,y)$ y está dada por

$$f(x,y) = \begin{cases} 255, & \text{si } |\nabla h(x,y)| \geq s, \\ 0, & \text{si } |\nabla h(x,y)| < s. \end{cases}$$

En la siguiente subsección se discute como elegir el umbral a partir de las características de la imagen en escala de grises.

2.2.2. Umbralización

Ahora que la imagen está en escala de grises, se puede pasar directamente a una binarización mediante el uso de un umbral s, con lo que se obtiene la imagen binarizada como

$$g(x,y) = \begin{cases} 255, & si\ f(x,y) > s, \\ 0, & si\ f(x,y) \leq s. \end{cases} \tag{2.2}$$

Cuando s es una constante aplicable sobre la imagen total, el proceso obtenido en (2.2) se refiere a un umbral global y cuando el valor de s cambia en una imagen se utiliza el término variable de umbral.
Si la intensidad de la distribución de objetos y píxeles de fondo son suficientemente distintos es posible usar un único umbral aplicable a la imagen de entrada. En el caso de estudio, cada imagen requiere un umbral determinado, ya que no necesariamente dos imágenes dadas tienen las mismas características de iluminación y de fondo de escena, por lo tanto, se requiere un algoritmo capaz de estimar el valor del umbral para cada imagen. El siguiente algoritmo iterativo se utilizó en el programa para este propósito [13]:

(1) Seleccionar un umbral inicial estimado global s.

(2) Segmentar la imagen usando s en (2.2), con lo que se obtienen dos grupos de píxeles G_1 y G_2. G_1 consiste de píxeles con valores de intensidad mayores a s y G_2 consiste de los píxeles con valores menores o iguales que s.

(3) Calcule el promedio de valores de intensidad m_1 y m_2 para los píxeles en G_1 y G_2 respectivamente.

(4) Calcule un nuevo valor de umbral como $s = \frac{1}{2}(m_1 + m_2)$.

(5) Repita los pasos (2)-(4) hasta que la diferencia entre dos valores de iteraciones sucesivas sean menor que un valor predeterminado Δs.

Otro enfoque para encontrar el umbral óptimo es el método de Otsu [13], además de que se pueden encontrar umbrales adaptativos locales a través de los métodos de Bernsen y Niblack [25].

2.2.3. Filtrado Gaussiano

Los filtros de suavizado espacial son usados para reducir las fuertes transiciones en intensidad de una imagen. Como el ruido aleatorio consiste de transiciones fuertes en la intensidad, una aplicación directa del suavizado es la reducción del ruido. La importancia reside en que un buen suavizado de la imagen nos proporciona una reducción de los detalles irrelevantes, donde irrelevante significa una región de píxeles que son pequeños respecto al tamaño del kernel. Por lo tanto, al aplicar un suavizado se puede obtener un mejor manejo de la información de la imagen para la extracción de regiones de interés.

Existen muchos filtros de suavizado, como el filtrado medio que considera una vecindad de píxeles y calcula la media de los valores encontrados y los guarda. En nuestro caso utilizaremos un kernel Gaussiano por sus distintas propiedades: es separable, es decir, se puede escribir como producto de dos funciones unidimensionales. Son isotrópicos, es decir, que la respuesta obtenida no depende de la orientación del kernel. El producto y la convolución de dos Gaussianas también son funciones Gaussianas.
Los kernels Gaussianos de la forma

$$G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}},\ K, s, t \in \mathbb{R},$$

son los únicos kernels simétricos circulares que son separables. Observe que s, t son reales, entonces tome $r^2 = s^2 + t^2$ y la expresión anterior se reescribe como

$$G(r) = Ke^{-\frac{r^2}{2\sigma^2}}.$$

Con esta transformación, la variable r es la distancia desde el centro a cualquier punto de la función G.

2.3. Morfología Matemática

El lenguaje de la morfología matemática es la teoría de conjuntos. La morfología ofrece un enfoque unificado y poderoso para

numerosos problemas de procesamiento de imágenes. Cuando se trabaja con imágenes, los conjuntos de morfología matemática representan objetos en esas imágenes. En el procesamiento de imágenes, se utiliza la morfología con dos tipos de conjuntos de píxeles: objetos y elemento estructurantes (SE). En general, los objetos se definen como conjuntos de píxeles de primer plano. Mientras que los elemento estructurantes se pueden especificar en términos de píxeles de primer plano y de fondo.
El enfoque basado en la morfología está relacionado con la forma o las características de una imagen. Esto implica eliminar las imperfecciones en la estructura de la imagen para disminuir el ruido que puede estar presente.
Se inicia la discusión morfología estudiando dos operaciones denominadas *erosión* y *dilatación*. Estas operaciones son fundamentales para el procesamiento morfológico.

2.3.1. Erosión y dilatación

La erosión se usa a menudo para eliminar detalles irrelevantes de la imagen binaria. Sea B el elemento estructurante, A un conjunto binario, entonces la operación de erosión denotada por $\ominus$, se define como:

$$A \ominus B = \{x \in A : B_x \subseteq A\},$$

donde $B_x = \{b + x : b \in B\}$. Esta expresión indica que la erosión de A por B es el conjunto de puntos x tales que B trasladado por x, está contenido en A.

La dilatación a menudo se usa para rellenar huecos de la imagen binaria. Sea B el elemento estructurante, A un conjunto binario, entonces la operación de dilatación denotada por $\oplus$, se define como:

$$A \oplus B = \{x \in A : B_x^s \cap A \neq \emptyset\},$$

donde $B^s = \{-b : b \in B\}$. Así, la dilatación de A por B es el conjunto de todos los desplazamientos, x, tales que los elementos de primer plano de B se superponen al menos con un elemento de A.

Por ejemplo, si se elige un elemento estructurante 3×3 con todas sus entradas con unos y se consideran los píxeles de la Figura 2.1, la operación de erosión se puede observar en la Figura 2.2 y la operación de dilatación en la Figura 2.3.

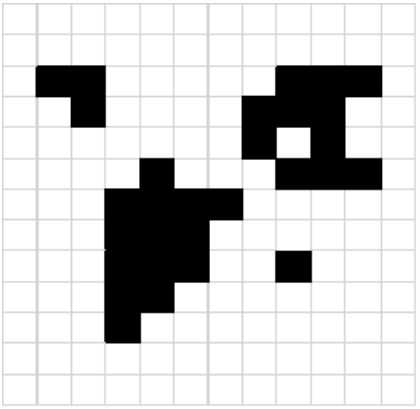

Figura 2.1: Subconjunto de píxeles.

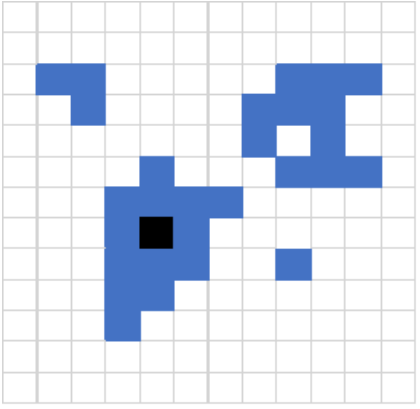

Figura 2.2: Operación de erosión con SE 3×3.

2.3.2. Apertura y cerradura

En esta sección se estudian otras dos operaciones morfológicas importantes que son la apertura y la clausura. Generalmente, la

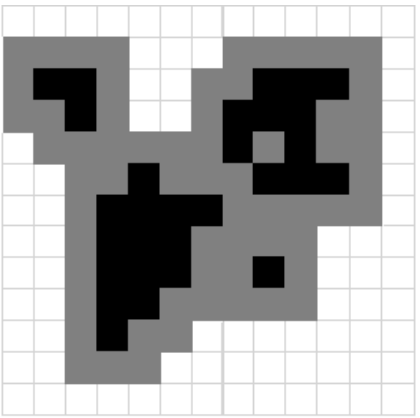

Figura 2.3: Operación de dilatación con SE 3×3.

apertura suaviza el contorno de un objeto, rompe franjas estrechas y elimina protuberancias delgadas. Por otro lado, la cerradura también ofrece suavizado de contorno, pero en general fusiona picos estrechos, elimina agujeros pequeños y rellena huecos en el contorno.
La apertura de un conjunto A por un elemento estructurante B, denotada por $A \circ B$ está definida como

$$A \circ B = (A \ominus B) \oplus B.$$

Por lo tanto, la apertura de A por B es el proceso resultante de la erosión de A por B seguido de dilatación por B.
La ecuación anterior, tiene una interpretación geométrica simple: La apertura de A por B es la unión de todas las traslaciones de B de modo que B se ajusta completamente en A. Con esto, podemos escribir:

$$A \circ B = \bigcup \{B_x | B_x \subseteq A\}.$$

Además, la apertura tiene las siguientes propiedades:

(i) $A \circ B$ es un subconjunto de A.

(ii) Si C es un subconjunto de D, entonces $C \circ B$ es un subconjunto de $D \circ B$.

(iii) $(A \circ B) \circ B = A \circ B$.

Ahora, la clausura de un conjunto A por un elemento estructurante B, denotado por $A \bullet B$, está definido como

$$A \bullet B = (A \oplus B) \ominus B.$$

El cuál nos dice que la cerradura de A por B es la dilatación de A por B, seguida por una erosión por B.

También, se puede ver la clausura como el complemento de la unión de todas las traslaciones de B que no se sobreponen a A. De este modo, la cerradura de A y B se puede escribir como

$$A \bullet B = \left[\bigcup \{B_x | B_x \cap A \neq \emptyset\}\right]^{\complement}.$$

Además, la apertura tiene las siguientes propiedades:

(i) A es un subconjunto de $A \bullet B$.

(ii) Si C es un subconjunto de D, entonces $C \bullet B$ es un subconjunto de $D \bullet B$.

(iii) $(A \bullet B) \bullet B = A \bullet B$.

Ahora, si se considera el subconjunto de píxeles de la Figura 2.1, la operación de apertura aplicada a este subconjunto de píxeles se muestra en la Figura 2.4, mientras que la aplicación de la cerradura se muestra en la Figura 2.5.

Para suprimir o atenuar elementos extraños que pueden ser muy brillantes, oscuros o simplemente para disminuir el ruido se puede utilizar un suavizado morfológico. Una forma de llevar a cabo el suavizado es realizar una apertura seguida de una clausura: $(A \circ B) \bullet B$ [13].

2.3.3. Esqueleto morfológico

Sea A un conjunto de píxeles, entonces el esqueleto de A, denotado por $S(A)$ se puede expresar en términos de erosión y aperturas, se puede mostrar que

$$S(A) = \bigcup_{k=0}^{K} S_k(A),$$

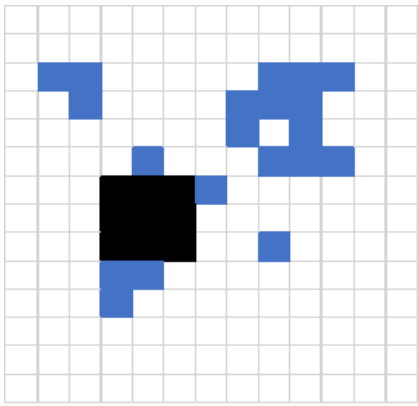

Figura 2.4: Operación de apertura con SE 3×3.

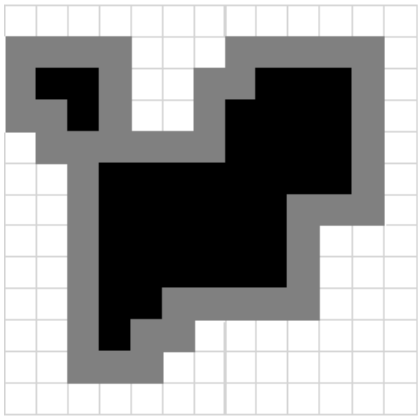

Figura 2.5: Operación de cerradura con SE 3×3.

donde $S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$, B es el elemento estructurante y $(A \ominus kB)$ indica k erosiones sucesivas iniciando con A, esto es, A es erosionado por B, el resultado es erosionado por B y así sucesivamente. Finalmente, K es el último paso iterativo antes de que A se erosione a un conjunto vacío, es decir,

$$K = \text{máx}\{k | \, (A \ominus kB) \neq \emptyset\}.$$

De este modo, $S(A)$ se puede obtener como la unión de los subconjuntos del esqueleto $S_k(A)$, $k = 0, 1, ..., K$.
También, se puede mostrar que A puede ser reconstruido de los

subconjuntos $S_k(A)$ como se muestra a continuación.

$$A = \bigcup_{k=0} (S_k(A) \oplus kB),$$

donde $(S_k(A) \oplus kB)$ denota k dilataciones sucesivas, iniciando con $S_k(A)$.

2.4. Modelos Ocultos de Markov

Los Modelos Ocultos de Markov son una extensión de las cadenas de Markov, en las que la producción de observaciones a lo largo de estados ya no es determinista sino que ocurre de acuerdo con una función probabilística de salida [10]. De este modo se puede decir que un Modelo Oculto de Markov consiste de dos procesos estocásticos, el primero es una cadena de Markov $(X_t)_{t\geq 1}$ a tiempo discreto, que se caracteriza por estados y probabilidades de transición. Mientras que el segundo proceso estocástico $(Y_t)_{t\geq 1}$ produce emisiones observables en cada momento, dependiendo de una distribución de probabilidad dependiente del estado. En este esquema, al tiempo que se evita un aumento en la complejidad de la cadena, se ofrece mayor libertad al proceso aleatorio, lo que permite capturar la variabilidad subyacente de secuencias discretas. La cadena de Markov sirve como una representación abstracta de las restricciones estructurales sobre la casualidad de los datos, además observe que los estados de la cadena no son visibles externamente, de ahí la denominación de *oculto*. Para el caso de estudio se requieren los MOM discretos, por lo que se revisan las problemáticas a las que nos enfrentamos al abordar estos modelos.

Definición 2.4.1. *Un MOM discreto es una quíntupla* (S, V, π, P, E) *donde*

- $S = \{s_1, s_2, \cdots, s_N\}$*, un conjunto de* N *estados, donde* X_t *es el estado al tiempo* t*.*
- $V = \{v_1, v_2, \cdots, v_M\}$ *un conjunto de* M *símbolos observados, donde* Y_t *es la observación al tiempo* t*.*

- $\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$ *con* $\pi_i = \mathbb{P}(X_1 = s_i)$*, la distribución de probabilidad inicial.*
- *La matriz de transiciones* $P = (p_{ij})$*, donde*

$$p_{ij} = \mathbb{P}(X_{t+1} = s_j | X_t = s_i),$$

$i, j = 1, 2, ..., N$, $t \geq 1$.

- $E = (e_i(k))$ *la matriz de emisiones, donde*

$$e_i(k) = \mathbb{P}(Y_t = v_k | X_t = s_i),$$

constituye la probabilidad de que el sistema estando en el estado s_i *genere la observación* v_k*.*

Para expresar un MOM de forma exacta y completa se necesitan especificar los parámetros del modelo N y M, los símbolos observados y las tres medidas de probabilidad P, E y π. En lo que sigue, se utilizará la notación $\lambda = (P, E, \pi)$ para indicar al conjunto de parámetros del modelo.

2.4.1. Los tres problemas básicos para MOMs

Antes de que los MOMs puedan utilizarse para aplicaciones reales, el desarrollo de un enfoque de un MOM implica abordar tres problemas y su respectiva solución. Sean $\lambda = (P, E, \pi)$ un modelo y $Y = \{Y_1 = y_1, Y_2 = y_2, \cdots, Y_T = y_T\}$ una secuencia de observaciones, los tres problemas son:

(1) Encontrar eficientemente la probabilidad

$$\mathbb{P}_\lambda(Y_1 = y_1, Y_2 = y_2, \cdots, Y_T = y_T),$$

es decir, calcular la probabilidad de la secuencia de observaciones Y dado el modelo λ.

(2) Encontrar la sucesión de estados ocultos más factible que pudo generar a Y.

(3) Estimar los parámetros de λ para maximizar

$$\mathbb{P}_\lambda(Y_1 = y_1, Y_2 = y_2, \cdots, Y_T = y_T).$$

El primer problema se conoce como problema de evaluación, el segundo como problema de decodificación y el tercero como problema de aprendizaje [10].
El problema 1 se puede ver como un problema de puntuar, es decir, que tan bien un modelo dado coincide con una secuencia de observación dada. Por ejemplo, si consideramos el caso en el que se está tratando de elegir el modelo que mejor se adapte a las observaciones [7], [8]. El problema 2 es aquel en el que se busca descubrir la parte oculta del modelo, es decir, encontrar la secuencia de estados *correcta*. Para situaciones prácticas usualmente se usa un criterio de optimalidad para resolver este problema lo mejor posible. Finalmente, en el problema 3 se desea optimizar una secuencia de variación de los parámetros del modelo. La secuencia de observación para ajustar los parámetros del modelo se denomina secuencia de entrenamiento ya que se utiliza para *entrenar* al MOM. El problema de entrenamiento es crucial para la mayoría de las aplicaciones de los MOMs ya que nos permite adaptar de manera óptima los parámetros del modelo a los datos de entrenamiento observados, es decir, crear los mejores modelos para los fenómenos reales.

2.4.2. Solución al primer problema

Continuando con la notación de la Definición 2.4.1, buscamos encontrar la probabilidad de la secuencia de observación $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$ dado el modelo λ, es decir, buscamos $\mathbb{P}_\lambda(Y)$. La forma más fácil de hacer esto es a través de enumerar toda secuencia posible de estados de longitud T, con lo que obtendríamos N^T posibles secuencias de estados de longitud T.
Considere una secuencia de estados fija $X = (X_1 = x_1, X_2 = x_2, ..., X_T = x_T)$, entonces la probabilidad de que ocurra la secuencia de estados X dado el modelo λ es

$$\mathbb{P}_\lambda(X) = \pi_{x_1} p_{x_1 x_2} p_{x_2 x_3} \cdots p_{x_{T-1} x_T}.$$

Ahora, si se supone independencia entre las observaciones, la probabilidad de la observación Y dada la secuencia de estados X y el

modelo λ está dada por

$$\mathbb{P}_\lambda(Y|X) = \prod_{t=1}^{T} \mathbb{P}_\lambda(Y_t = y_t|X_t = x_t) = \prod_{t=1}^{T} e_t(y_t).$$

De este modo, la probabilidad de que X y Y ocurran simultáneamente, es el producto de los dos términos anteriores, esto es,

$$\mathbb{P}_\lambda(Y, X) = \mathbb{P}_\lambda(Y|X)\mathbb{P}_\lambda(X).$$

Por lo tanto, la probabilidad de Y dado el modelo λ se obtiene sumando la distribución conjunta sobre todas los posibles secuencias de estados X, resultando que

$$\begin{aligned}\mathbb{P}_\lambda(Y) &= \sum_X \mathbb{P}_\lambda(Y|X)\mathbb{P}_\lambda(X) \\ &= \sum_{(x_1,\ldots,x_T)} \pi_{x_1} e_{x_1}(y_1) p_{x_1x_2} e_{x_2}(y_2) \cdots p_{x_{T-1}x_T} e_{x_T}(y_T).\end{aligned} \tag{2.3}$$

La interpretación de la ecuación anterior es la siguiente: al tiempo $t = 1$, nos encontramos en el estado x_1 con probabilidad π_{x_1} y se genera el símbolo y_1 con probabilidad $e_{x_1}(y_1)$. Al tiempo $t = 2$, se hace una transición del estado x_1 al estado x_2 con probabilidad $p_{x_1x_2}$ y se genera el símbolo y_2 con probabilidad $e_{x_2}(y_2)$. Este proceso continúa, hasta el tiempo $t = T$, cuando se hace una transición del estado x_{T-1} al estado x_T con probabilidad $p_{x_{T-1}x_T}$ y se genera el símbolo y_T con probabilidad $e_{x_T}(y_T)$.
Por otra parte, calcular (2.3) implica cálculos de orden $2TN^T$, lo cual no es computacionalmente factible incluso para valores pequeños de N y T. Por lo tanto, se requiere de un procedimiento más eficiente para resolver el problema 1, una solución es el procedimiento forward.

Considere

$$\alpha_t(i) = \mathbb{P}_\lambda\left(Y_1 = y_1, Y_2 = y_2, \cdots, Y_t = y_t, X_t = i\right),$$

conocida como variable forward que representa la probabilidad conjunta de obtener la secuencia de observaciones parciales ($Y_1 =$

$y_1, Y_2 = y_2, ..., Y_t = y_t)$ hasta el tiempo t y que el proceso se encuentre en el estado i al tiempo t, dado el modelo λ. El procedimiento forward es el siguiente:

(i) Inicialización: $\alpha_1(i) = \pi_i e_i(y_i)$, $i = 1, 2, \cdots, N$.

(ii) Recursión:

$$\alpha_{t+1}(j) = e_j(y_{t+1}) \sum_{i=1}^{N} \alpha_t(i)\, p_{ij}, \;\; j = 1, 2, ..., N, \;\; t = 1, 2, ..., T-1.$$

(iii) Terminación: $\mathbb{P}_\lambda(Y_1 = y_1, Y_2 = y_2, \cdots, Y_T = y_T) = \sum_{i=1}^{T} \alpha_T(i)$.

El paso (i) inicializa las probabilidades forward como la probabilidad conjunta de encontrarse en el estado i y observación inicial y_1. Para el paso (ii), el producto $\alpha_t(i)p_{ij}$ es la probabilidad conjunta de que $(Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t)$ sea observada y el estado j se alcanza en el tiempo $t+1$ a través del estado i al tiempo t. Sumando este producto sobre todos los posibles estados (N) en el tiempo t da como resultado la probabilidad de j al tiempo $t+1$ con todas las observaciones parciales previas. Una vez hecho esto y con j conocido, $\alpha_{t+1}(j)$ es obtenido contabilizando la observación y_{t+1} en el estado j, es decir, multiplicando la cantidad sumada por la probabilidad $e_j(y_{t+1})$. El cálculo se realiza para todos los estados dado un tiempo t y se itera para $t = 1, 2, ..., T-1$. Finalmente, el paso (iii) nos da el cálculo de $\mathbb{P}_\lambda(Y)$ como la suma de las variables terminales forward $\alpha_T(i)$, debido a que por definición $\alpha_T(i) = \mathbb{P}_\lambda(Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T, X_T = i)$ y por lo tanto, $\mathbb{P}(Y)$ es la suma de las $\alpha_T(i)$.
Bajo este enfoque, se requieren cálculos de orden de N^2T, en vez de $2TN^T$ como lo requiere el cálculo directo.

De forma similar, podemos considerar la variable dada por

$$\beta_t(i) = \mathbb{P}_\lambda(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T | X_t = i),$$

conocida como variable backward y que representa la probabilidad de obtener una secuencia de observación parcial $(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, ..., Y_T = y_T)$ desde el instante $t+1$ hasta el instante T, dado que el estado es i en el tiempo t y dado el modelo λ. El procedimiento backward es el siguiente:

(i) Inicialización: $\beta_T(i) = 1,\ i = 1, 2, ..., N.$

(ii) Recursión:

$$\beta_t(i) = \sum_{j=1}^{N} p_{ij}\beta_{t+1}(j)e_j(y_{t+1}),\ i = 1, 2, ..., N,\ t = T-1, T-2, ..., 1.$$

(iii) Terminación:

$$\mathbb{P}_\lambda\left(Y_1 = y_1, Y_2 = y_2, \cdots, Y_T = y_T\right) = \sum_{i=1}^{T} \beta_1(i)\pi_i e_i(y_1).$$

En el paso (i) se define $\beta_T(i) = 1$ para todo i. El paso (ii) muestra que para encontrarse en el estado i en el tiempo t y tener en cuenta la secuencia de observación desde el tiempo $t + 1$ en adelante, se deben considerar todos los posibles estados j en tiempo $t + 1$, teniendo en cuenta la transición de i a j, así como la observación y_{t+1} en el estado j y luego tener en cuenta la secuencia de observación parcial restante del estado j. Al igual que en el procedimiento forward, el cálculo requiere un orden de N^2T.
De esta manera los procedimientos forward y backward nos ayudan a resolver el problema 1 de manera eficiente, pero también nos ayudarán para dar solución al problema 2 y 3.

2.4.3. Solución al segundo problema

Hay muchas formas posibles de resolver el problema 2, ya que debemos encontrar la secuencia de estados *óptima* asociada con la secuencia de observación. La dificultad, recae en la definición de estado óptimo, pues hay una amplia variedad de criterios de optimalidad. Por ejemplo, un posible criterio de optimalidad es elegir los estados x_t que son más probables individualmente en cada instante de tiempo. Este criterio de optimalidad maximiza el número esperado de estados individuales correctos. Definamos la variable

$$\gamma_t(i) = \mathbb{P}_\lambda(X_t = i|Y),$$

llamada variable de probabilidad posterior que representa la probabilidad de encontrarse en el estado i al tiempo t, dada la secuencia de observación Y y el modelo λ.

Observe que $\gamma_t(i)$ se puede expresar como

$$\begin{aligned}\gamma_t(i) &= \mathbb{P}_\lambda(X_t = i|Y) \\ &= \frac{\mathbb{P}_\lambda(X_t = i, Y)}{\mathbb{P}_\lambda(Y)} \\ &= \frac{\mathbb{P}_\lambda(X_t = i, Y)}{\sum_{j=1}^{N} \mathbb{P}_\lambda(X_t = j, Y)}.\end{aligned}$$

Usando las variables forward y backward se tiene que

$$\mathbb{P}_\lambda(X_t = i, Y) = \alpha_t(i)\beta_t(i),$$

entonces se puede escribir

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}.$$

Ahora, con ayuda de $\gamma_t(i)$, podemos encontrar el estado individual, digamos x_t*, con máxima probabilidad de haber generado una observación concreta al momento t como

$$x_t* = \underset{1\leq i\leq N}{\arg\max} \left[\gamma_t(i)\right].$$

Aunque la ecuación anterior maximiza el número esperado de estados correctos, eligiendo el estado más probable para cada t, podría haber algunos problemas con la secuencia de estados resultante. Por ejemplo, cuando el MOM tiene transiciones de estados que tienen probabilidad cero, es decir, $p_{ij} = 0$, para algunos i, j, la secuencia de estados óptima podría ser una secuencia de estados no válida. Lo anterior, ocurre ya que esta ecuación simplemente determina el estado más probable en cada instante, sin tener en cuenta la probabilidad de ocurrencia de una secuencia de estados. En algunas aplicaciones, el criterio más utilizado es encontrar la mejor secuencia de estado (trayectoria), esto es, maximizar $\mathbb{P}_\lambda(X|Y)$ lo cual es equivalente a maximizar $\mathbb{P}_\lambda(X, Y)$. Existe una técnica formal para encontrar esta única secuencia de estados óptima, basada en métodos de programación dinámica, llamada algoritmo de Viterbi.
Para encontrar la mejor secuencia de estados

$$X = (X_1 = x_1, X_2 = x_2, ..., X_T = x_T),$$

dada una secuencia de observaciones

$$Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T),$$

necesitamos definir

$$\delta_t(i) = \max_{x_1, x_2, \cdots, x_{t-1}} \mathbb{P}_\lambda \left(X_1 = x_1, \cdots X_t = i, Y_1 = y_1, \cdots, Y_t = y_t\right),$$

la probabilidad de la mejor secuencia de estados que finaliza en el estado i teniendo en cuenta las primeras t observaciones.
Además, se cumple que

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i) p_{ij}\right] e_j\left(y_{t+1}\right).$$

Para recuperar la secuencia de estados, necesitamos realizar un seguimiento del argumento que maximizó la expresión anterior. Para esto se usa el arreglo ψ. El procedimiento completo para encontrar la mejor secuencia de estados se puede establecer como sigue.

- Inicialización:

$$\begin{aligned} \delta_1(i) &= \pi_i e_i\left(y_1\right), \ i = 1, ..., N. \\ \psi_1(i) &= 0. \end{aligned}$$

- Recursión:

$$\delta_{t+1}(j) = \max_{1 \leq i \leq N} \left[\delta_t(i) p_{ij}\right] e_j\left(y_{t+1}\right) \text{ y } \psi_{t+1}(j) = \arg\max_{1 \leq i \leq N} \left[\delta_t(i) p_{ij}\right],$$

$t = 1, 2, ..., T-1, \ j = 1, 2, ..., N.$

- Terminación: $x_T^* = \arg\max_{1 \leq i \leq N} \left[\delta_T(i)\right]$.

- Reconstrucción: $x_t^* = \psi_{t+1}\left(x_{t+1}^*\right), \ t = T-1, T-2, ..., 1.$

2.4.4. Solución al tercer problema

El tercer problema de MOMs es por mucho el más difícil, pues se trata de determinar un método para ajustar los parámetros del modelo P, E y π para satisfacer un cierto criterio de optimización. No se conoce una forma analítica de estimar un conjunto

de parámetros para el modelo que maximice la probabilidad de la secuencia de observación de forma cerrada. Sin embargo, se puede elegir $\lambda = (P, E, \pi)$ de tal manera que la probabilidad, $\mathbb{P}_\lambda(Y)$, es localmente maximizada usando un procedimiento iterativo como el método de Baum-Welch [7] y [8]. Iniciamos con el procedimiento de reestimación de los parámetros, definiendo

$$\xi_t(i,j) = \mathbb{P}_\lambda(X_t = i, X_{t+1} = j|Y),$$

esto es, la probabilidad de estar en el estado i al tiempo t y en el estado j al tiempo $t+1$ dada la observación Y y el modelo λ. Usando las variables forward y backward podemos escribir $\xi_t(i,j)$ como

$$\begin{aligned}\xi_t(i,j) &= \frac{\mathbb{P}_\lambda(X_t = i, X_{t+1} = j, Y)}{\mathbb{P}_\lambda(Y)} \\ &= \frac{\alpha_t(i)p_{ij}e_i(y_{t+1})\beta_{t+1}(j)}{\mathbb{P}_\lambda(Y)} \\ &= \frac{\alpha_t(i)p_{ij}e_i(y_{t+1})\beta_{t+1}(j)}{\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_t(k)p_{kl}e_k(y_{t+1})\beta_{t+1}(l)}.\end{aligned}$$

En el problema 2, se definió $\gamma_t(i)$ como la probabilidad de estar en el estado i en el momento t dada la secuencia Y y el modelo λ, por lo tanto, podemos relacionar $\gamma_t(i)$ con $\xi_t(i,j)$ sumando sobre j, esto es

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j).$$

Además, si sumamos sobre el tiempo t conseguimos una cantidad que puede ser interpretada como el número esperado de veces que el estado i es visitado o bien el número esperado de transiciones hechas desde el estado i, esto es

$$\sum_{t=1}^{T-1}\gamma_t(i).$$

De manera similar, la suma de $\xi_t(i,j)$ sobre t puede ser interpretada como el número esperado de transiciones del estado i al estado j, esto es,

$$\sum_{t=1}^{T-1}\xi_t(i,j).$$

Usando las ecuaciones anteriores, podemos obtener un método para la reestimación de los parámetros de un MOM. Se supone un modelo de inicio $\lambda = (P, E, \pi)$, se calculan las variables forward y backward con el fin de calcular $\xi_t(.,.)$ y $\gamma_t(.)$. Las siguientes ecuaciones se conocen como fórmulas de reestimación y se utilizan para actualizar los parámetros del MOM.

(i) $\hat{\pi}_i = \gamma_1(i)$,

(ii) $\hat{p}_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$,

(iii) $\hat{e}_j(k) = \dfrac{\sum_{t=1}^{T} \gamma_t(j)_{y_t = v_k}}{\sum_{t=1}^{T} \gamma_t(j)}$.

Podemos interpretar (i) como el número de veces que el proceso se encuentra en el estado i en el momento $t = 1$. Interpretamos (ii) como el cociente del número esperado de transiciones del estado i al estado j entre el número esperado de transiciones desde el estado i. Finalmente, (iii) se interpreta como el cociente del número esperado de veces en las que el proceso se encuentra en el estado j observando el símbolo v_k entre el número de veces en las que el proceso se encuentra en el estado j.
Si definimos un modelo $\lambda = (P, E, \pi)$ y definimos el modelo reestimado $\hat{\lambda} = (\hat{P}, \hat{E}, \hat{\pi})$ con ayuda de (i)-(iii), Baum probó [8] que el modelo λ define un punto crítico de la función de verosimilitud, que en su caso $\hat{\lambda} = \lambda$ o bien que el modelo $\hat{\lambda}$ es más probable que el modelo λ en el sentido que $\mathbb{P}_{\hat{\lambda}}(Y) > \mathbb{P}_{\lambda}(Y)$, es decir, encontramos un modelo nuevo $\hat{\lambda}$ para el cual la secuencia de observación es más probable que se haya producido. De esta manera, se obtiene el siguiente procedimiento.

1. Elegir distribuciones de probabilidad P, E y π tales que $\lambda = (P, E, \pi)$.
2. Calcular (i), (ii) y (iii).
3. Hacer $\hat{\lambda} = (\hat{P}, \hat{E}, \hat{\pi})$.
4. Si $\hat{\lambda} = \lambda$ o $\mathbb{P}_{\hat{\lambda}}(Y) > \mathbb{P}_{\lambda}(Y)$ terminar: como salida $\hat{\lambda}$. En otro caso, tome $\lambda = \hat{\lambda}$ y regresar al paso 2.

Capítulo 3

Etapas del reconocimiento

En este capítulo, se revisa el procedimiento planteado para el reconocimiento automático de placas a través de los MOMs. Para esto, haremos uso de las herramientas del Capítulo 2. El procedimiento planteado en este trabajo, requiere algunos requisitos para la imagen de entrada: la imagen debe estar en formato BMP de 24 bits, debe tener buena calidad y debe estar tomada de frente con tal de que los caracteres no presenten una inclinación significativa. Cabe destacar que los algoritmos planteados pueden recibir la imagen completa con el auto y ruido de fondo o la imagen recortada al tamaño de la placa. En la Sección 3.1 se aborda el preprocesamiento y la segmentación de la imagen de entrada, que resulta básico para las siguientes etapas, incluyendo el reconocimiento. En la Sección 3.2 se dedica al procedimiento propuesto para la ubicación de la placa. Después, se revisa el proceso de reconocimiento de los caracteres de la placa con ayuda de los MOMs. Finalmente, se muestran los resultados de los Algoritmos.

3.1. Preprocesamiento y segmentación de la imagen de entrada

Recordamos que el formato de la imagen de entrada al sistema de reconocimiento es el formato BMP. La primera etapa es ubicar

la sección de los datos y las dimensiones de la imagen dada, lo que se logra con la información obtenida en el Capítulo 1. Ubicadas las dimensiones y los píxeles de la imagen, guardamos las dimensiones en dos variables m y n, donde m es la altura y n el ancho. Como la imagen está descrita en el sistema RGB, cada píxel de la imagen de entrada está representada por 8 bits para el color rojo, 8 bits para el color verde y 8 bits para el color azul, lo que supone tener 3 variables por píxel tomando los valores $0, 1, ..., 255$. Para reducir el tamaño de la información, hacemos el cambio a escala de grises por medio de la expresión (2.1), con lo que se logra obtener una variable por píxel, pues bajo esta transformación $R = G = B$ con valores de $0, 1, ..., 255$. La información de estos píxeles la guardamos en la una matriz A, de tamaño $m \times n$ con entradas $a_{ij} \in \{0, 1, ..., 255\}$ para todo $i = 1, 2, ..., m$ y $j = 1, 2, ..., n$. Considere la imagen de entrada que se muestra en la Figura 3.1, entonces la imagen en escala de grises resultante se muestra en la Figura 3.2.

Figura 3.1: Imagen de entrada.

Figura 3.2: Imagen en escala de grises.

En cuanto a la etapa de segmentación, lo que sigue es aplicar un suavizado a los datos de la imagen en escala de grises, este suavizado se realiza con un filtro Gaussiano y la información de los píxeles se guardan en una matriz B, de tamaño $m \times n$. Ahora, la imagen en escala de grises suavizada se puede observar en la Figura 3.3.

Figura 3.3: Imagen en escala de grises suavizada.

Después, se harán dos procesos, el primero es pasar a una imagen binaria mediante el algoritmo proporcionado en la Sección

2.2.2 y la información generada es guardada en la matriz A, de esta manera, $a_{ij} = 0$ o $a_{ij} = 255$, $i = 1, 2, ..., m$ y $j = 1, 2, ..., n$. Esta información nos será de ayuda para rescatar los caracteres de la placa. La imagen de salida después de aplicar la umbralización se muestra en la Figura 3.4. El segundo proceso, tiene la finalidad de iniciar con la ubicación de la placa. Aquí, se combina la detección de bordes vertical (Sección 2.2.1) con el concepto de umbralización. Ahora, la información resultante se guarda en una matriz C de tamaño $m \times n$ que tiene entradas $c_{ij} = 0$ o $c_{ij} = 255$, $i = 1, 2, ..., m$ y $j = 1, 2, ..., n$. Realizando estos pasos, la imagen de salida se muestra en la Figura 3.5.

Figura 3.4: Imagen binaria.

Figura 3.5: Imagen binaria después de aplicar la detección de bordes verticales.

Observe que la eficiencia en el algoritmo de umbralización hace posible tener una imagen binaria nítida en la región de la placa, lo que será de ayuda en etapas posteriores. Con esto, finaliza la etapa de preprocesamiento de la imagen, lo que sigue es utilizar los datos resultantes (la información almacenada en las matrices A y C) para continuar con las siguientes etapas .

3.2. Ubicación de la placa

La ubicación de la placa es un paso clave para el reconocimiento de los caracteres pues de no obtener la región que contiene todos estos, se obtendría una salida de caracteres totalmente equivocada. Se han realizado muchos esfuerzos por obtener la ubicación exacta de la placa ya que se presentan diversos problemas al intentar obtener tal región.

La ubicación de la placa se puede obtener mediante una secuencia de pasos que incluyen detección de bordes verticales en conjunto con umbralización, y un suavizado con la aplicación del algoritmo ARS, que fue motivado por los inconvenientes encontrados en varias bibliografías [12], [21], [24].
Debido a la presencia de los caracteres en la región de interés, la

mayor parte de los bordes verticales son encontrados aquí. Para la detección de los bordes se aplica el gradiente vertical Sobel a la imagen en escala de grises en conjunto con la umbralización. Lo que resulta en una imagen segmentada que resalta de manera considerable los caracteres de la placa, Figura 3.5.
El siguiente paso es suavizar los datos obtenidos en la Figura 3.5, con este suavizado se obtiene una reducción del tamaño de los datos lo que implica un mejor manejo de la información y una mayor eficiencia computacional. El suavizado se lleva acabo al aplicar las funciones propuestas en los Algoritmos 1 y 2. Estas funciones fueron motivadas por el estudio de las operaciones morfológicas de erosión y dilatación. De hecho, el Algoritmo 1 está inspirado en la operación de erosión, mientras que el Algoritmo 2 está inspirada en la operación de dilatación, ambas están basadas en un elemento estructurante 1×5. A diferencia de la erosión y la dilatación, se considera únicamente el centro de cada elemento estructurante para ampliar o reducir la información. Ambas funciones tienen como argumentos a C y sus dimensiones m y n, donde C como antes es la información de la imagen ya preprocesada. Como resultado, se obtiene una imagen que contiene menor información relevante, pero la ventaja más significativa es que en los caracteres de la placa las regiones verticales son las más grandes, Figura 3.6. Por lo tanto, nos interesamos en esas regiones, en cuanto a cercanía, y por lo tanto se obtiene una región estimada a partir de las coordenadas obtenidas en el algoritmo ARS. Para delimitar de manera correcta la región de la placa, se realiza un escaneo de izquierda a derecha para encontrar el ancho de la placa y de arriba hacia abajo para obtener su altura. Tal región es nuestra región candidata, como se observa en la Figura 3.7.

Algorithm 1: Algoritmo 1

Datos $C = (c_{ij})$, m, n;
Resultado $B = (b_{ij})$;
for $0 \leq i < m - 4$ **do**
  for $0 \leq j < m - 4$ **do**
    if $c_{ij} = 255$ *y* $c_{ij+1} = 255$ *y* $c_{ij+2} = 255$ *y* $c_{ij+3} = 255$ *y* $c_{ij+4} = 255$ **then**
      $b_{ij} = 0$;
      $b_{ij+1} = 0$;
      $b_{ij+2} = 255$;
      $b_{ij+3} = 0$;
      $b_{ij+4} = 0$;
    else
      $b_{ij} = 0$;
      $b_{ij+1} = 0$;
      $b_{ij+2} = 0$;
      $b_{ij+3} = 0$;
      $b_{ij+4} = 0$;

Algorithm 2: Algoritmo 2

Datos $C = (c_{ij})$, m, n;
Resultado $B = (b_{ij})$;
for $0 \leq i < m - 4$ **do**
  for $0 \leq j < m - 4$ **do**
    if $c_{ij+2} = 255$ **then**
      $b_{ij} = 255$;
      $b_{ij+1} = 255$;
      $b_{ij+2} = 255$;
      $b_{ij+3} = 255$;
      $b_{ij+4} = 255$;
    else
      $b_{ij} = 0$;
      $b_{ij+1} = 0$;
      $b_{ij+2} = 0$;
      $b_{ij+3} = 0$;
      $b_{ij+4} = 0$;

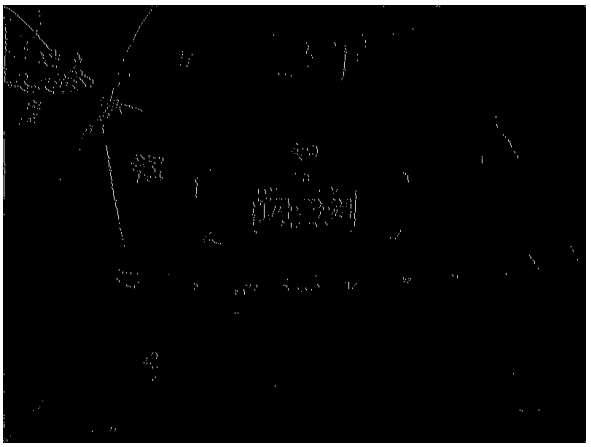

Figura 3.6: Imagen después de aplicar suavizado de los datos.

Figura 3.7: Región de la placa.

Finalmente, se presenta el Algoritmo ARS en el Algoritmo 3. Observe que como salida se obtiene un vector de cuatro entradas, donde (b_0, b_1) es la esquina superior izquierda y (b_2, b_3) es la esquina inferior derecha de la región de la placa.

Algorithm 3: Algoritmo ARS

Datos $A = (a_{ij})$, m, n;
Resultado $b = (b_0, b_1, b_2, b_3)$;
$A = EscalaGrises(A)$;
$A = FiltroGaussiano(A)$;
$A = DetectaBordes(A)$;
$A = Binarizar(A)$;
$A = Algoritmo1(A, m, n)$;
$A = Algoritmo2(A, m, n)$;
$A = Algoritmo2(A, m, n)$;
$A = Algoritmo1(A, m, n)$;
N: Número de píxeles distintos de 0 que tiene A;
Se obtienen las coordenadas de los píxeles distintos de cero de A y se guardan en un archivo de texto B;
$b = Coor(B, m, n, N)$;
donde $Coor()$ es la función que se muestra en el Algoritmo 4.

Algorithm 4: Algoritmo 4, función $Coor()$.

Datos $B = (b_{ij})$, $i = 0, 1$, $j = 0, 1, ..., N-1$, m, n, N;
Resultado $b = (b_0, b_1, b_2, b_3)$;
Se registra como k_i el número de píxeles que contiene la columna i-ésima de B;
Si k_i es mayor que 10 se considera como una columna de importancia, en otro caso se descarta;
Se verifica cuántos píxeles son consecutivos. Si hay una conexión de al menos 5 píxeles se guardan sus coordenadas y longitud, en otro caso se descartan;
Para la elección de la región, se obtiene el promedio de longitud de los segmentos mayores a 5 y se verifica si los segmentos de interés están a una distancia respecto de la media de a lo más un cuarto del ancho y alto de la imagen;
Luego, se obtienen las coordenadas extremales de entre los segmentos elegidos, es decir, se buscan las coordenadas de los segmentos que están en la parte superior izquierda y en la parte inferior derecha, "los más alejados".

3.3. Reconocimiento de los caracteres con MOMs

El reconocimiento de caracteres es la tarea más importante en el reconocimiento de la placa del vehículo, pero depende de etapas previas, por eso la importancia de las etapas del preprocesamiento de la imagen. Existen diferentes técnicas para el reconocimiento de caracteres como el uso de plantillas para comparar, uso de redes neuronales, chain code y Modelos Ocultos de Markov [11]. Para cada imagen de entrada con vehículo, el tamaño de la región que contiene la placa podría diferir y por lo tanto, para cada imagen los caracteres tendrán diferente tamaño, forma y estilo lo que nos podría encaminar a realizar falso reconocimientos, lo que se traduce en una falla del sistema de detección. Por esta razón, la región de cada carácter se normaliza a un tamaño de 12×24 píxeles y luego se extrae el vector de características como el ángulo central de los caracteres de cada región segmentada. Nos enfocamos en el reconocimiento de caracteres a través de los Modelos Ocultos de Markov.

Como se ha revisado en la Sección 2.4, el MOM es una técnica de coincidencia de patrones probabilísticos que tiene la capacidad de absorber tanto variabilidad como similitud entre las imágenes. Con ayuda de los MOMs, consideramos los caracteres de la placa como una secuencia de estados $X = (X_1 = x_1, X_2 = x_2, ..., X_T = x_T)$. En cada estado, podemos generar un vector de observación $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$, según la distribución de probabilidad asociada. Los parámetros del MOM se entrenan usando un vector de observación extraído de una muestra representativa de la imagen de la placa. Finalmente, el reconocimiento de una imagen de placa desconocida se basa en la probabilidad de que el MOM genere una imagen de la placa.

En este trabajo, una imagen de placa bidimensional se convierte en un vector de características de una dimensión. Inicialmente la placa se segmenta por carácter, después de cada región que contiene un carácter se extraen 4 características de ángulo central. El vector de características forma un vector de observación uni-

dimensional que será utilizado en las etapas de entrenamiento y validación.

Para llevar a cabo el reconocimiento, se requiere una base de datos. Nuestra base de datos consta de 40 placas, por lo que los datos se encuentran restringidos al hablar de las letras, recordando que las placas del estado de Puebla constan de 3 letras y 4 números, pues muchas letras se repiten. Sin embargo, se puede aprovechar la información obtenida por los dígitos que parece ser suficiente para iniciar con el trabajo de entrenamiento. Por lo anterior, se muestra que el algoritmo de reconocimiento es útil para dígitos y a futuro al ampliar la base de datos se podrá mostrar la efectividad en el reconocimiento de letras.

Dado el vector de observación Y, el objetivo es buscar la clase W que maximice la probabilidad $\mathbb{P}(W|Y)$. En este caso, se busca que cada clase contenga las características de un carácter fijo. Con ayuda de los MOMs, W está representada implícitamente por su modelo subyacente $\mathbb{P}(W|Y) = \mathbb{P}_\lambda(Y)$. Para el reconocimiento de caracteres se construye un MOM λ_i para cada clase y usamos el algoritmo forward para estimar $\mathbb{P}_{\lambda_i}(Y)$ para $i = 1, ..., N_c$, donde N_c es el número de clases. En este caso, $N_c = 33$, ya que en las placas del estado de Puebla, se tienen contempladas 23 letras del alfabeto y los 10 dígitos. Finalmente, el algoritmo identifica λ_i que maximiza $\mathbb{P}_{\lambda_i}(Y)$, para $i = 1, ..., N_c$.

Por otro lado, existen algunos inconvenientes que están involucrados al hablar de los MOMs. Algunos de los problemas que mencionan las referencias [7], [8] y [10] son: la elección de la topología de los MOMs, la estimación inicial de los parámetros y el escalamiento que requieren los algoritmos forward, backward y por lo tanto, los algoritmos que nos resuelven los tres problemas que están relacionados con los MOMs, Subsección 2.4.1. Para elegir la topología del MOM, nos basamos en [17] y para el escalamiento se sigue el proceso dado en [7]. Para la estimación inicial de los parámetros, la literatura propone usar el algoritmo EM o el método segmental k-means [10]. En este trabajo, se propone inicializar la matriz de emisiones con respecto a la mínima distancia de la

observación a cada una de las 4 clases propuestas. La matriz de transiciones se inicializa aleatoriamente y el vector de probabilidades iniciales se inicializa de manera uniforme.

Iniciamos con el algoritmo de extracción de características, el cuál se establece de la siguiente manera y fue motivado por [17]. Después de obtener la región de la placa, se localiza la región sobre la imagen en escala de grises, es decir, la matriz B que se guardó al aplicar el filtro Gaussiano. Después, se obtiene el esqueleto morfológico de toda la región de la placa, Figura 3.8. Luego, para extraer las características que se necesitan en el reconocimiento, se realiza una segmentación de los caracteres con lo que cada subimagen contendrá un solo carácter de la placa. Con lo anterior se obtienen 7 regiones como se muestra en la Figura 3.9. Ahora, cada región que contiene un carácter se normaliza a una imagen de 12 píxeles, Figuras 3.10 - 3.16. Posteriormente, se aplica un adelgazamiento del carácter por medio del algoritmo planteado en [22], Figuras 3.17 - 3.23. A continuación, cada región que contiene el carácter adelgazado o simplemente esqueleto, se divide en 4 celdas de acuerdo al siguiente procedimiento.

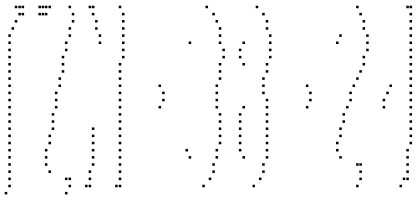

Figura 3.8: Esqueleto morfológico de la placa.

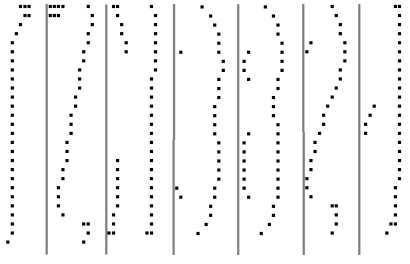

Figura 3.9: Segmentación del esqueleto morfológico de la placa.

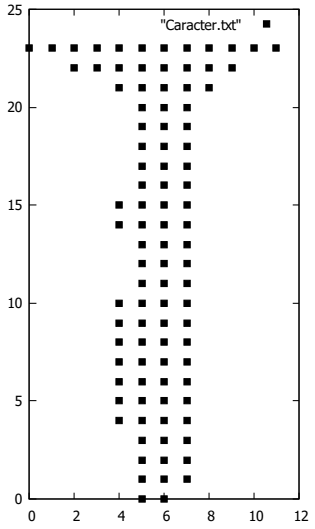

Figura 3.10: Carácter normalizado a 12×24 píxeles.

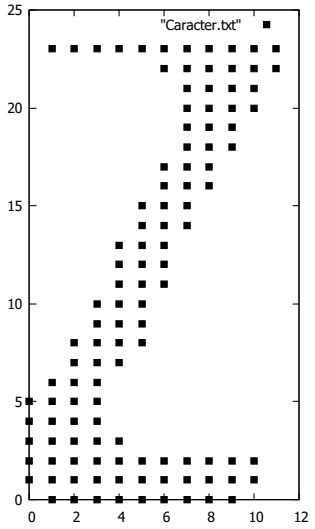

Figura 3.11: Carácter Z normalizado a 12×24 píxeles.

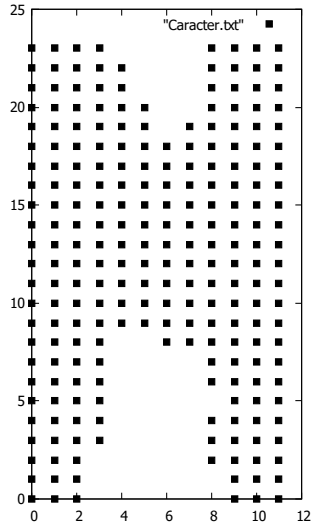

Figura 3.12: Carácter M normalizado a 12×24 píxeles.

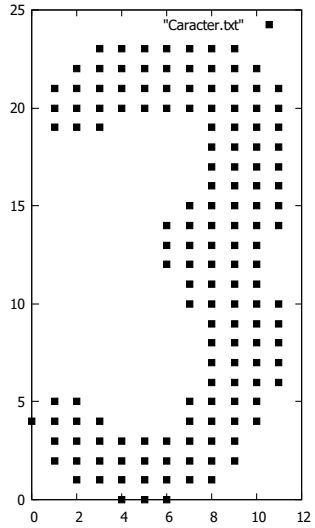

Figura 3.13: Dígito 3 normalizado a 12×24 píxeles.

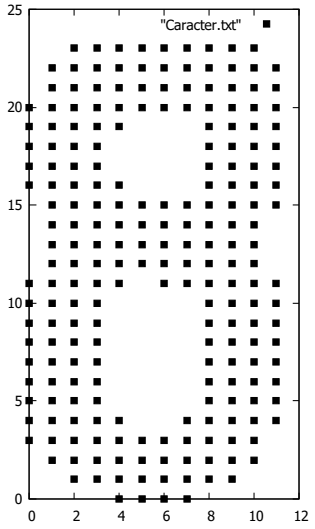

Figura 3.14: Dígito 8 normalizado a 12×24 píxeles.

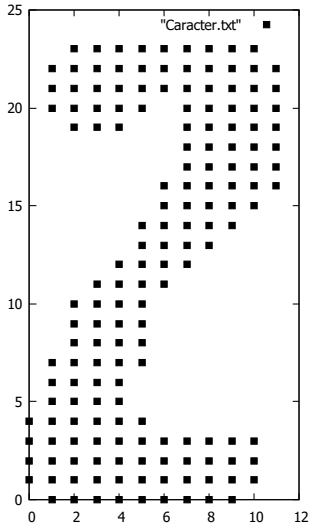

Figura 3.15: Dígito 2 normalizado a 12×24 píxeles.

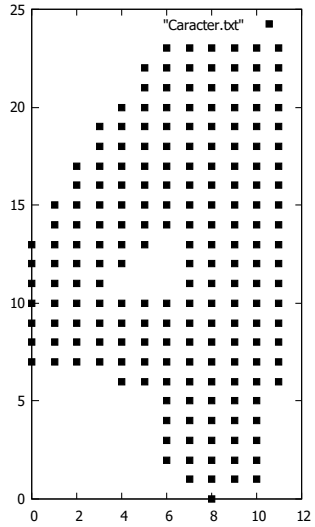

Figura 3.16: Dígito 4 normalizado a 12×24 píxeles.

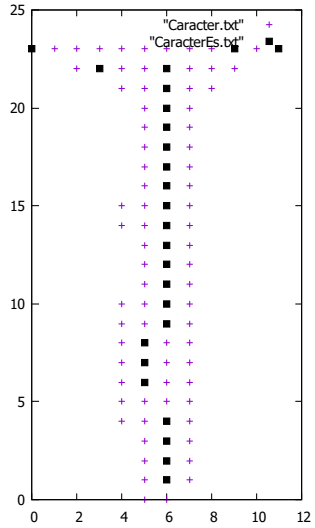

Figura 3.17: Esqueleto carácter T.

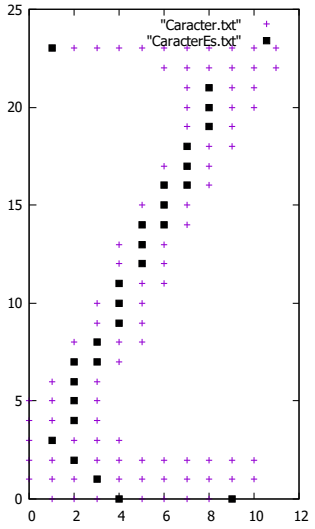

Figura 3.18: Esqueleto carácter Z.

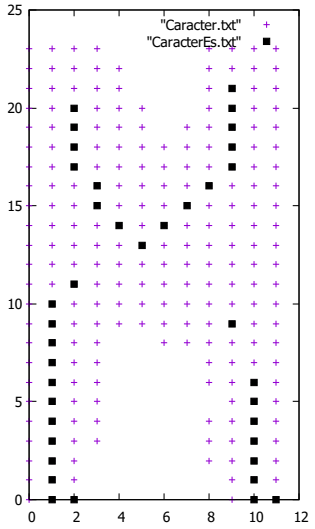

Figura 3.19: Esqueleto carácter M.

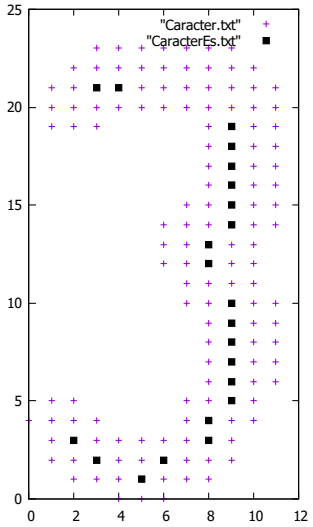

Figura 3.20: Esqueleto dígito 3.

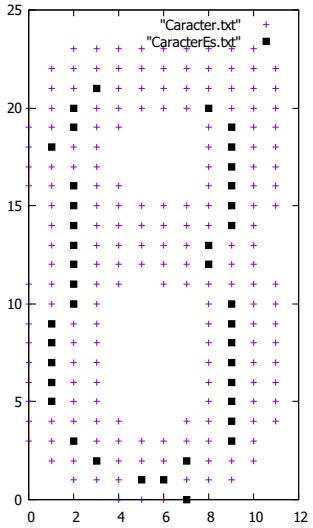

Figura 3.21: Esqueleto dígito 8.

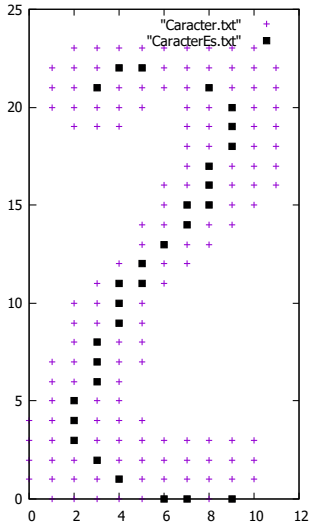

Figura 3.22: Esqueleto dígito 2.

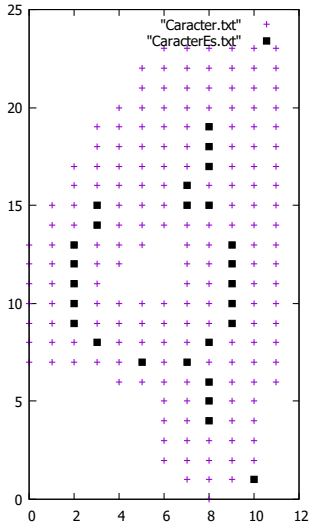

Figura 3.23: Esqueleto dígito 4.

- Para cada región que contiene un esqueleto de un carácter calcule el centro de gravedad de la imagen mediante la ex-

presión

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \text{ y } \hat{y} = \frac{1}{N}\sum_{j=1}^{N} y_j, \tag{3.1}$$

donde N es el número de píxeles sobre el esqueleto y (x_i, y_i) es el i-ésimo píxel del esqueleto.

- A través de los puntos $\hat{x}$, $\hat{y}$ haga una división vertical en cada región, con lo que obtenemos dos bloques.
- Ubique el centro de cada uno de los dos bloques usando (3.1).
- A través de los puntos $\hat{x}$, $\hat{y}$ haga una división horizontal en cada región, con lo que se obtienen 4 celdas, Figuras 3.24 - 3.30.

Finalmente, ubique el centro de cada una de las celdas usando (3.1). En cada celda, calcule el ángulo que forma cada punto central con la esquina inferior derecha de la celda. Los ángulos centrales de todas las celdas de imagen constituyen el conjunto de características para la imagen de la placa.

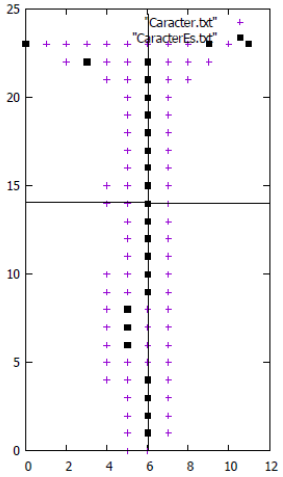

Figura 3.24: División carácter T.

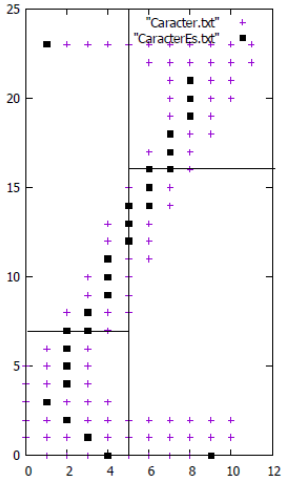

Figura 3.25: División carácter Z.

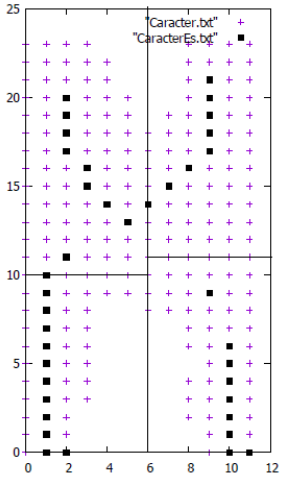

Figura 3.26: División carácter M.

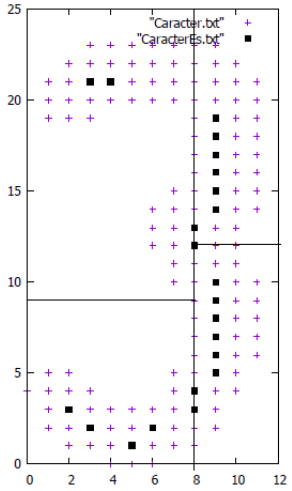

Figura 3.27: División dígito 3.

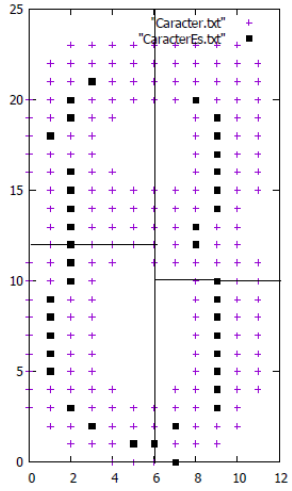

Figura 3.28: División dígito 8.

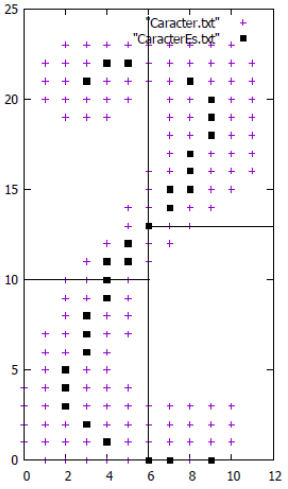

Figura 3.29: División dígito 2.

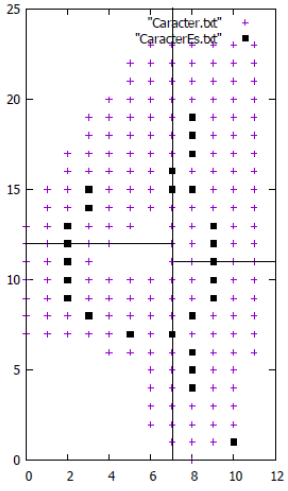

Figura 3.30: División dígito 4.

Para el entrenamiento, cada una de las imágenes de la placa se modela mediante la estimación de los parámetros de MOM para un conjunto dado de observaciones. Se utilizan 4 imágenes de cada carácter para entrenar cada MOM. De cada carácter de la placa se extraen 4 características, para formar el vector de observación.

Los parámetros se eligen en función de un criterio que maximiza la probabilidad de los datos de observación Y. Esta maximización se realiza utilizando el algoritmo de Baum-Welch, Sección 2.4.4. El proceso de entrenamiento sigue los siguientes pasos:

(1) Inicializar el MOM (P, E, π).
Cada uno de los caracteres de entrenamiento muestreados se segmenta en 4 estados y los vectores de observación de estas placas son usados para obtener la estimación inicial de la matriz de emisiones E. Los valores iniciales para P se establecen dado el modelo de izquierda a derecha de la topología del MOM, como se observa en la Figura 3.31 y los valores de las transiciones se consideran aleatorias. Finalmente, π se inicializa como una distribución uniforme.

(2) Se reestiman los parámetros del modelo utilizando el algoritmo Baum-Welch para maximizar $\mathbb{P}_\lambda(Y)$.
El procedimiento iterativo se detiene cuando la diferencia entre los puntajes de probabilidad de la iteración actual $k+1$ y los del anterior k es menor que un umbral preestablecido ϵ, esto es,

$$|\mathbb{P}_{\lambda_{k+1}}(Y) - \mathbb{P}_{\lambda_k}(Y)| < \epsilon.$$

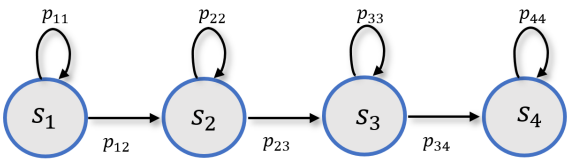

Figura 3.31: Topología del MOM para el carácter de la placa.

El Algoritmo ARM para el reconocimiento se muestra en el Algoritmo 5.

Algorithm 5: Algoritmo ARM

Datos $A = (a_{ij}),\ m\ n,$
$b = (b_0, b_1, b_2, b_3), Y = (y_0, y_1, y_2, y_3)$;
Resultado $\lambda_j,\ j = 1, 2, ..., N_c$;
Se ubican las coordenadas obtenidas en ARS en la imagen Binarizada A.;
Se guarda la región encontrada en una matriz $Z = (z_{kl})$;
$Z = EsqueletoMorfologico(Z)$;
$Z = SegmentarCaracteres(Z)$;
Cada región $Z_i,\ i = 0, 1, ..., 6$, se normaliza a un tamaño de 12×24 píxeles.;
Para cada $Z_i,\ i = 0, 1, ..., 6$, se extrae su esqueleto;
$a_i = ExtraeCarac(Z_i),\ i = 0, 1, ..., 6$;
$EntrenamientoMOM(a_i),\ i = 0, 1, ..., 6$;
Inicializar $\lambda_j = (P_j, E_j, \pi_j),\ j = 1, 2, ..., N_c$;
Reestimación de parámetros usando el algoritmo Baum-Welch;
Se regresa la clase que maximiza $\mathbb{P}_{\lambda_j}(Y),\ j = 1, 2, ..., N_c$;

Como se hace en la etapa de entrenamiento, los vectores de características extraídos de cada uno de los estados de la imagen de la placa de prueba se utiliza para formar el vector de observación. Los MOMs entrenados se utilizan para calcular la función de probabilidad de la siguiente manera.

- Considere a $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$ como la secuencia de observación basada en ángulo central generada a partir de la imagen que contiene el carácter de la placa para ser reconocida.
- La probabilidad del vector observado dado cada modelo de la imagen de carácter $\mathbb{P}_{\lambda_i}(Y)$ se calcula utilizando el algoritmo forward.
- El vector observado se etiqueta con el modelo de clase que maximiza $\mathbb{P}_{\lambda_i}(Y)$.
- La imagen de carácter de la placa se reconoce como imagen de un carácter de la placa en la base de datos si $\mathbb{P}_{\lambda_k}(Y) = \text{máx}\{\mathbb{P}_{\lambda_i}(Y) | i = 1, 2, ..., N_c\}$.

3.4. Resultados

Recordamos que para inicializar los parámetros del MOM, la matriz de transición se inicializa de manera aleatoria, respetando la topología planteada en la Figura 3.31 y la distribución inicial de manera uniforme. Las filas de la matriz de emisiones representan los 4 estados en que se particionó la imagen del carácter y las columnas representan las medias de los valores de las 4 imágenes que consideramos para cada una de las clases involucradas: $0, 1, ..., 9$. Al inicializar las matrices de emisión para el primer dígito de la placa en la Figura 3.7 se obtuvieron las matrices $E_0, E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8$ y E_9.

$$E_0 = \begin{bmatrix} 0.1 & 0.2 & 0.4 & 0.3 \\ 0.1 & 0.3 & 0.2 & 0.4 \\ 0.1 & 0.4 & 0.2 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

$$E_1 = \begin{bmatrix} 0.1 & 0.3 & 0.2 & 0.4 \\ 0.1 & 0.4 & 0.2 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

$$E_2 = \begin{bmatrix} 0.1 & 0.4 & 0.2 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

$$E_3 = \begin{bmatrix} 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}.$$

$$E_4 = \begin{bmatrix} 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}.$$

$$E_5 = \begin{bmatrix} 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.4 & 0.1 & 0.3 \end{bmatrix}.$$

$$E_6 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

$$E_7 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.3 \end{bmatrix}.$$

$$E_8 = \begin{bmatrix} 0.2 & 0.4 & 0.1 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

$$E_9 = \begin{bmatrix} 0.4 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.4 & 0.3 & 0.1 & 0.2 \\ 0.4 & 0.3 & 0.1 & 0.2 \end{bmatrix}.$$

De forma similar se realiza la inicialización de los 10 modelos para cada uno de los dígitos restantes. Después del entrenamiento, se obtuvo que el primer dígito era 3, el segundo era 8, el tercero era 2 y el último era 4, lo que en efecto corresponde a los dígitos de la placa, Figura 3.7.

En conclusión, se logró obtener el algoritmo ARM que utiliza los MOMs en la etapa de entrenamiento para reconocer los dígitos de la placa. Este algoritmo consta de varias etapas, la primera consiste en ubicar las coordenadas de la placa en la imagen de entrada, las cuales se obtienen con ayuda del algoritmo ARS. Después, se extrae el esqueleto morfológico de la placa, lo que permite la segmentación de los caracteres. Seguido de la segmentación, se realiza una normalización que consiste en cambiar las dimensiones originales del carácter a un tamaño de 12×24 píxeles. Ahora, a cada carácter normalizado se le extrae su esqueleto y en base a él se obtiene el vector de características que será introducido en la etapa de entrenamiento del MOM. En el entrenamiento, para cada dígito en la placa, inicializamos un modelo $\lambda_j = (P_j, E_j, \pi_j)$, $j = 1, 2, ..., 10$, que representan las 10 clases de dígitos disponibles

y posteriormente, reestimamos los parámetros de cada modelo con ayuda del algoritmo Baum-Welch. Finalmente, se devuelve la clase que maximiza la probabilidad de la observación dada.

Capítulo 4

Conclusiones

En la búsqueda de un reconocimiento de placas con un enfoque de MOMs, nos enfrentamos a varios problemas: Ubicar la placa y realizar su reconocimiento. Primero, se logró ubicar la región de la placa a través del Algoritmo ARS con ayuda de las funciones propuestas que fueron motivadas de las operaciones morfológicas de erosión y dilatación. Con este algoritmo, logramos reducir los datos de la imagen de manera considerable con tal de rescatar la información básica para ubicar la placa. Después, se logró hacer un reconocimiento de los caracteres con ayuda de los MOMs, a través del algoritmo propuesto ARM. En este algoritmo, redimensionamos los caracteres a un tamaño de 12×24 píxeles y posteriormente aplicamos un adelgazamiento a cada carácter para extraer el vector de observación y con este entrenamos el MOM.
Los resultados muestran un gran potencial en el reconocimiento para las letras de la placa. Sin embargo, se requiere de una base de datos más amplia con la finalidad de mejorar el reconocimiento ya que al realizar ensayos del programa se ha visto que el trabajo de entrenamiento falla al identificar el dígito 1 con el dígito 4 y el dígito 0 con el dígito 8. Por esta razón, trabajos futuros contemplan ampliar o conseguir una base de datos con el fin de realizar el reconocimiento de placas completo. Además, se puede abordar el problema de reconocimiento de placas quitando algunos de los requisitos impuestos para la imagen de entrada de este trabajo. También, se podría utilizar una forma alternativa de extracción de características que ayude a mejorar el sistema de reconocimiento.

Bibliografía

[1] Borko Furht, Esad Akar and Whitney Angelica Andrews, *Digital Image Processing: Practical Approach*, Springer, 2018.

[2] Hongbing Wang, Shuqi Wei, Rong Huang, Shuai Deng, Fei Yuan, Anjun Xu and Jiani Zhou, *Recognition of Plate Identification Numbers Using Convolution Neural Network and Character Distribution Rules*, pp. 2044-2051.ISIJ International, 2019.

[3] Hu Hongping and Bai Yanping, *A Kind of License Plate Location Based on Mathematical Morphology and Edge Detection*, 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, pp. 2291-2294, 2011.

[4] John Miano, *Compressed image file formats : JPEG, PNG, GIF, XBM, BMP*, Addison Wesley Longman, 1999.

[5] John Van Der Hoek and Robert J. Elliott, *Introduction to Hidden Semi-Markov Models*, Cambridge University Press, 2018.

[6] Jun-Wei Hsieh, Shih-Hao Yu and Yung-Sheng Chen, *Morphology-based license plate detection in images of differently illuminated and oriented cars*, Journal of Electronic Imaging 11(4), pp. 507–516, 2002.

[7] L. R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceeding of the IEEE, Vol 77, pp. 257-286, 1998.

[8] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[9] Meera Ramadas and Ajith Abraham, *Metaheuristics for Data Clustering and Image Segmentation*, Springer, 2019.

[10] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu and Ching Y. Suen, *Character Recognition Systems, A Guide for Students and Practioners*, Jhon Wiley & Souns, 2007.

[11] Nuzulha Khilwani Ibrahim, Emaliana Kasmuri, Norazira A Jalil, Mohd Adili Norasikin and Mohamad Riduwan Md Nawawi, *License Plate Recognition (LPR): A Review with Experiments for Malaysia Case Study*, arXiv preprint arXiv:1401.5559, 2014.

[12] Priyanka Prabhakar and P. Anupama, *A Novel Design For Vehicle License Plate Detection and Recognition*, pp. 7-12, 2014.

[13] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Pearson, Fourth Edition, 2018.

[14] Rajshekhar Mukherjee, Amit Pundir, Dharmendra Mahato, Gaurav Bhandari and Geetika Jain Saxena, *A Robust Algorithm for Morphological, Spatial Image-Filtering and Character Feature Extraction and Mapping Employed for Vehicle Number Plate Recognition*, IEEE WiSPNET 2017 conference, pp. 864-869, 2017.

[15] Robert J. Elliott, Lakhdar Aggoun and John B.Moore, *Hidden Markov Models, Estimation and control*, Springer, 1995.

[16] Rohit M. Thanki and Ashish M. Kothari, *Digital Image Processing using SCILAB*, Springer, 2019.

[17] S. Adebayo Daramola, E. Adetiba, A. U. Adoghe, J. A. Badejo, I. A Samuel and T. Fagorusi, *Automatic Vehicle Identification System Using License Plate*, International Journal of Engineering Science and Technology (IJEST), 2011.

[18] Samy Bakheet and Ayoub Al-Hamadi, *Chord-Length Shape Features For License Plate Character Recognition*, Springer Science+Business MEDIA, 2020.

[19] Sandeep Angara and Melvin Robinson, *License Plate Character Recognition Using Binarization and Convolutional Neural Networks*, Springer Nature Switzerland, pp. 272–283, 2020.

[20] Sérgio Montazzolli and Claudio Jung, *Real-Time Brazilian License Plate Detection and Recognition Using Deep Convolutional Neural Networks*, IEEE, pp. 55-62, 2017.

[21] Shu-jian Maa and Jian-yu Zhao, *A method of license plate location based on mathematical morphology and corner detection*, 2011 International Conference on Network Computing and Information Security, pp. 257-260, 2011.

[22] T. Y. Zhang and C. Y. Suen, *A Fast Parallel Algorithm for Thinning Digital Patterns*, Communications of the ACM 27, pp. 236-239, 1984.

[23] Tran Duc Duan, Tran Le Hong Du, Tran Vinh Phuoc and Nguyen Viet Hoang, *Building an Automatic Vehicle License-Plate Recognition System*, Intl. Conf. in Computer Science, pp. 21-24, 2005.

[24] Wasif Shafaet Chowdhury, Ashikur Rashid Khan and Jia Uddin, *Vehicle License Plate Detection Using Image Segmentation and Morphological Image Processing*, Springer International Publishing, pp. 142-154, 2018.

[25] Wilhelm Burger and Mark J. Burge, *Digital Image Processing*, Springer Verlag, Second Edition, 2016.

[26] Zhiwen Wang and Li Shaozi, *Research and Implement for Vehicle License Plate Recognition Based on improved BP Network*, 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering, pp. 101-104, 2010.