CAPÍTULO 3. DESARROLLO

3.1 Breve Descripción

El reconocimiento de imágenes se lleva a cabo generalmente en dos etapas: Etapa de Entrenamiento y Etapa de Reconocimiento.

En términos generales, el entrenamiento es la etapa en la que se procesa la imagen, se detecta y extrae el contorno del objeto de interés, y se describe con alguna herramienta matemática, como cadenas de códigos, descriptores de Fourier, descriptores de regiones basados en longitud, descriptores de regiones basados en área, etc., siendo en este caso la Transformada de Distancia. Al finalizar el entrenamiento se tiene una variable que contiene la información que representa al objeto dentro del sistema.

El reconocimiento es la etapa que procesa la imagen y compara algunos de los elementos a reconocer con la o las variables conocidas. El producto final anuncia que el objeto ha sido reconocido o no.

3.2 Notas sobre Matlab

Existen dos tipos de archivos de Matlab y son: archivos de escritura y funciones. Los archivos de escritura son secuencias de declaraciones de comandos o subprogramas de Matlab. Las variables de un archivo de escritura son globales y modifican el valor de las variables del mismo nombre en el ambiente de la sesión actual de MATLAB.

Los archivos de función extienden MATLAB. Se pueden crear funciones nuevas específicas a un problema y que tengan el mismo estado que otras funciones de Matlab. Las variables en un archivo de la función son locales inicialmente; sin embargo, se pueden declarar globales de ser necesario. El archivo de función debe iniciar forzosamente con una fórmula que indique su nombre y argumentos de entrada y de salida, como se muestra a continuación:

function [variable(s) de salida]= nombre del archivo[variable(s) de entrada]

En ambos tipos de archivo, el nombre del archivo es el nombre del comando que deber ser ingresado en la ventana de comandos de Matlab para ejecutar la rutina.

3.3 Algoritmo

3.3.1 Etapa de Entrenamiento

Detección de Contornos

1. Leer la imagen.
2. Convertir la imagen RGB en imagen de intensidad.
3. Obtener el contorno de los objetos que se encuentran en la imagen. Métodos: *canny, bwperim, convolución*.

Morfología

4. Procesar morfológicamente la imagen con los siguientes operadores:
    a. Cerrar
    b. Rellenar

c. Remover

d. Engrosar

Extracción de Objetos

5. Extraer el contorno del objeto de interés con intervención del usuario. Método: *bwselect* (comando de Matlab que permite la intervención del usuario, una vez indicado el objeto, lo extrae automáticamente).

Representación de Objetos por Transformada de Distancia

6. Crear una matriz de distancia. Matriz en la cual los píxeles del contorno valen 0 y los demás 255.
    a. Se lleva a cabo un barrido en la matriz original.
    b. Cambiar los valores con el comando “if” y asignaciones.
7. Aplicar el operador distancia a la matriz de distancia.
    a. Comparar la distancia de los píxeles con un primer barrido de izquierda a derecha.
    b. Comparar la distancia de los píxeles con un segundo barrido de derecha a izquierda.
8. Crear una variable (arreglo) que contenga la transformada del objeto.
9. Identificar la variable con una etiqueta asignada por el usuario.

3.3.2 Etapa de Reconocimiento

Detección de Contornos

1. Ingresar y leer la imagen.
2. Convertir la imagen RGB en imagen de intensidad.
3. Obtener el contorno de los objetos que se encuentran en la imagen.

Morfología

4. Procesar morfológicamente la imagen con los siguientes operadores:
    a. Cerrar
    b. Rellenar
    c. Remover
    d. Engrosar

Extracción de Objetos

5. Extraer el contorno del objeto de interés con intervención del usuario

Reconocimiento de Objetos por Transformada de Distancia

6. Obtener la distancia promedio del contorno de la nueva matriz al contorno de la(s) matriz(ces) conocida(s) mediante un mapeo entre la matriz nueva y el arreglo de los objetos ya conocidos.
7. Concluir:

    a. Si la distancia promedio en la comparación con el primer arreglo (primer objeto conocido) es menor a las demás distancias promedio, mostrar mensaje: “el objeto se reconoce, es un(a) [etiqueta dada por el usuario para el primer objeto]”

    b. Si la distancia promedio en la comparación con el segundo arreglo (segundo objeto conocido) es menor a las demás distancias promedio, mostrar mensaje: “el objeto se reconoce, es un(a) [etiqueta dada por el usuario para el segundo objeto]”

    c. Si la distancia promedio en la comparación con el tercer arreglo (tercer objeto conocido) es menor a las otras distancias promedio, mostrar mensaje: “el objeto se reconoce, es un(a) [etiqueta dada por el usuario para el tercer objeto]”

    d. Si la distancia promedio en la comparación con el cuarto arreglo (cuarto objeto conocido) es menor a las otras distancias promedio, mostrar mensaje: “el objeto se reconoce, es un(a) [etiqueta dada por el usuario para el cuarto objeto]”

3.4 Código

3.4.1 Niveles del programa

El usuario del programa debe ejecutar el comando “entrenamiento” en la línea de comando de Matlab para iniciar el entrenamiento. El comando ejecuta 4 subprogramas encargados de ingresar las Transformadas de Distancia de 4 objetos respectivamente. La Figura 3.1 muestra esquemáticamente éste proceso.

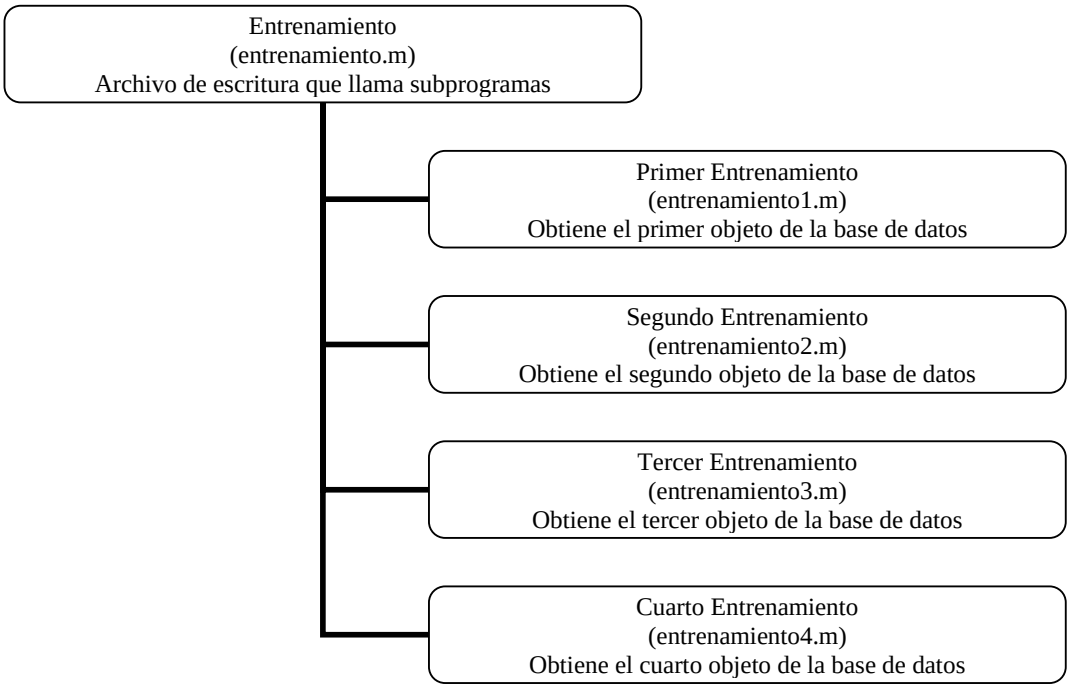

Figura 3.1. Subprogramas de Entrenamiento

En la etapa de reconocimiento el usuario debe ingresar una imagen con la siguiente sintaxis I=(‘nombre de la imagen . formato’) e ingresar el comando reconocimiento(I). *Reconocimiento* también llama 3 subprogramas que llevaran a cabo el proceso, desde el procesamiento de la imagen de entrada hasta la conclusión si el objeto es o no reconocido.

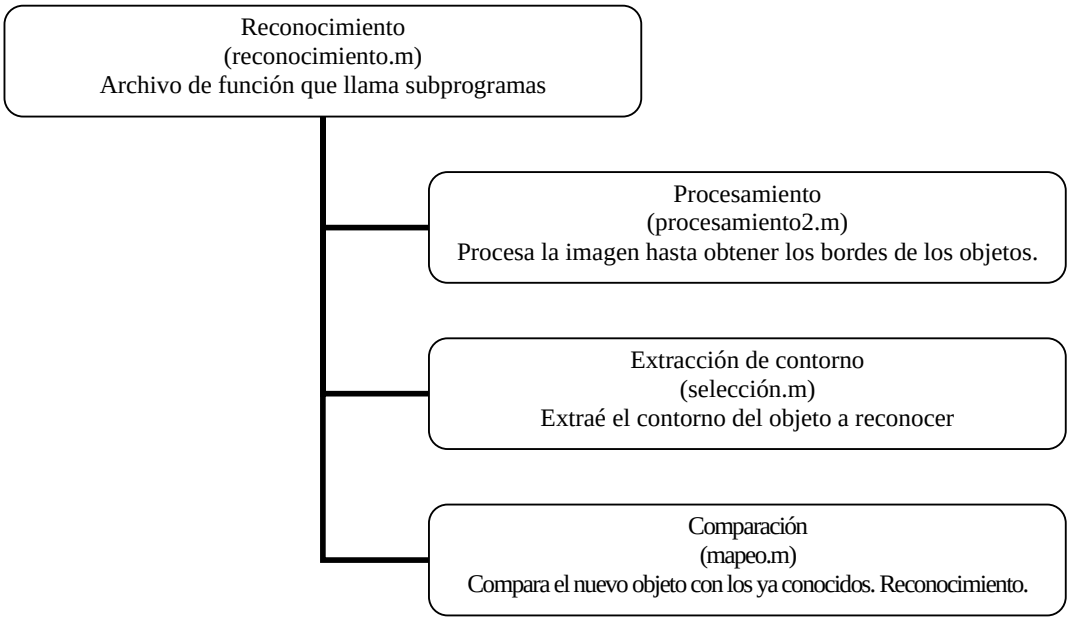

Figura 3.2. Subprogramas de Reconocimiento

3.4.2 Descripción y Funcionamiento

En el siguiente texto, el formato del código se identifica con itálicas. Dentro del código, el símbolo '%' se emplea para indicar textos comentados.

Cuando el usuario ingresa el comando entrenamiento en la ventana de comando de Matlab se llama al archivo entrenamiento.m, el cual es un archivo de escritura que ejecuta cuatro subprogramas, uno por cada objeto que quiere darse a conocer al sistema.

(entrenamiento.m)

%entrenamiento

entrenamiento1

close all

entrenamiento2

close all

entrenamiento3

close all

entrenamiento4

close all

A continuación se describe el primer subprograma que se ejecuta, que es entrenamiento1.

(entrenamiento1.m)

%entrenamiento1

global matriz1

global objeto1

gobal es el comando con el cual se declaran globales las variables. Matlab maneja variables locales en cada archivo de función y las aísla de las variables que se utiliza en el espacio de trabajo y de las variables de otras funciones. Sin embargo, si varias funciones, e incluso el espacio de trabajo declaran una variable como global, entonces todos compartirán una copia de esa variable. En éste caso, las variables a compartir son matriz1 y objeto1. Matriz1 es la matriz generada por la Transformada de Distancia del primer objeto de la base de datos. Objeto1 es el nombre que el usuario da al objeto que se ingresa al sistema, es una etiqueta de identificación.

I1=imread('foto1.jpg');

El comando *imread* lee la imagen a partir de un archivo gráfico. Si la imagen que se lee es una imagen de intensidad, I1 es un arreglo bidimensional, Si el archivo es una imagen RGB, I1 es un arreglo tridimensional. Los formatos de imagen que puede leer Matlab son los siguientes:

- Joint Photographic Experts Group (JPEG)
- Tagged Image File Format (TIFF)
- Graphics Interchange Format (GIF)
- Windows Bitmap (BMP)
- Portable Network Graphics (PNG)
- Hierarchical Data Format (HDF)
- Windows Paintbrush (PCX)
- X Window Dump (XWD)
- Windows Cursor resources (CUR)
- Windows Icon resources (ICO)

La imagen a leer debe estar ubicada dentro del directorio C.\matlabR12\toolbox\images\ para tener acceso a ella siguiendo la forma 'nombre.formato'. Sin embargo es posible acceder a ella desde el espacio de trabajo si en el argumento del comando *imread* se especifica la ruta donde se encuentra la imagen.

Foto1.jpg es una imagen RGB que se muestra en la Figura 3.3.

Figura 3.3. Foto1.jpg

imshow(I1)

title('Imagen Original')

imshow es el comando que se utiliza para mostrar una imagen en una ventana separada de la mesa de trabajo. *imshow* soporta imágenes binarias, de intensidad y RGB. El comando *title* añade un título a la imagen mostrada.

En la ejecución del programa, los comandos mencionados muestran la matriz I1 como la Figura 3.4.

I2=I1(:,:,1);

Es una de las instrucciones más comunes para convertir una imagen RGB en imagen de intensidad, reduce la matriz tridimensional I1 en una matriz bidimensional I2. El resultado gráfico de ésta conversión se aprecia en la Figura 3.5.

Figura 3.4. Imagen mostrada por el comando imshow

Figura 3.5. Imagen de intensidad

E=edge(I2,'canny');

E2=bwperim(E);

El comando *edge* encuentra el borde de la imagen de intensidad. Toma una imagen de intensidad como entrada (I2) y regresa una imagen binaria (E) del mismo tamaño que la primera y con 1's donde la función encuentra un borde en I2 y 0's en cualquier otro píxel.

El argumento *'canny'* es uno de los 6 métodos disponibles para obtener el contorno; el método fue explicado en el capítulo 2.2.1.

bwperim determina el perímetro de objetos en imágenes binarias, éste comando regresa una imagen binaria que contiene solo los píxeles pertenecientes al perímetro de un objeto en la imagen de entrada (E). *bwperim* considera que un píxel es parte del perímetro si es 1 y hay al menos un píxel con valor 0 en su vecindad, la cual puede ser de 4 o de 8 vecinos conectados. La Figura 3.6 ejemplifica la explicación mostrando la imagen de la variable E2.

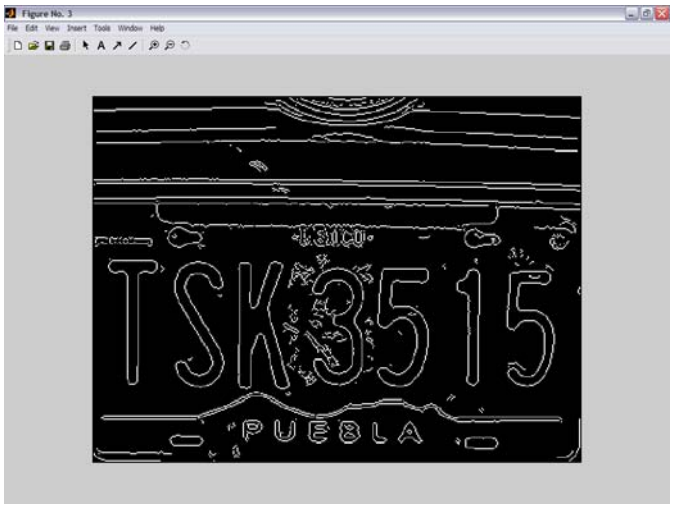

Figura 3.6. Ejemplo de los comandos *edge* y *bwperim*

```
B2=bwmorph(E2, 'close');
B=bwfill(B2, 'holes');
B1=bwmorph(B, 'remove');
B3=bwmorph(B1, 'thicken');
figure,imshow(B3)
title('Contornos de la Imagen')
```

El comando *bwmorph* lleva acabo operaciones morfológicas en imágenes binarias, en este caso realiza las operaciones: '*close*' (cerrar) que se muestra en la Figura 3.7(a), '*remove*' (remover), mostrado en la Figura 3.7(c) y '*thicken*' (engrosar) sobre las variables que se indican. El operador "thicken" se utiliza con la finalidad de facilitar al usuario la selección del contorno siguiente.

Así mismo, se emplea el comando denominado *bwfill*, cuya función es llenar (cambiar a 1) las regiones de fondo de las imágenes binarias. Con el argumento '*holes*', se rellenan los huecos en la imagen binaria B1. El comando determina automáticamente los píxeles dentro de los huecos de los objetos y cambia el valor de éstos de 0 a 1. La Figura 3.7(b) ejemplifica la forma en que el operador modifica la imagen.

```
%seleccion
BS=bwselect(B3, 8);
figure, imshow(BS)
title('Contorno seleccionado')
```

El comando *bwselect* permite seleccionar objetos dentro de una imagen binaria, en este caso, el comando muestra la imagen de B3 en pantalla y permite seleccionar unas coordenadas de referencia con el ratón. Se marcan coordenadas con el botón izquierdo del ratón. Si se presiona <BACKSPACE> ó <DELETE> se eliminan los puntos ya seleccionados. El punto final se selecciona con doble clic ó con un clic derecho. Al presionar <RETURN> se termina la selección sin haber agregado puntos. La Figura 3.8 los contornos de interés que se pueden obtener de la fotografía de la Figura 3.4.

Figura 3.7. Procesamiento Morfológico: (a) Operador “close”. (b) Operador “fill”. (c) Operador remove. (d) Operador “thicken”.

```
%matriz de distancias

M=zeros(size(BS));

[r,c]=size(BS);

for j=1: r,

   for i=1: c,

      pixel=BS(j,i);

      if (pixel==0)

        M(j,i)=255;

      else M(j,i)=0;

      end

  end

end
```

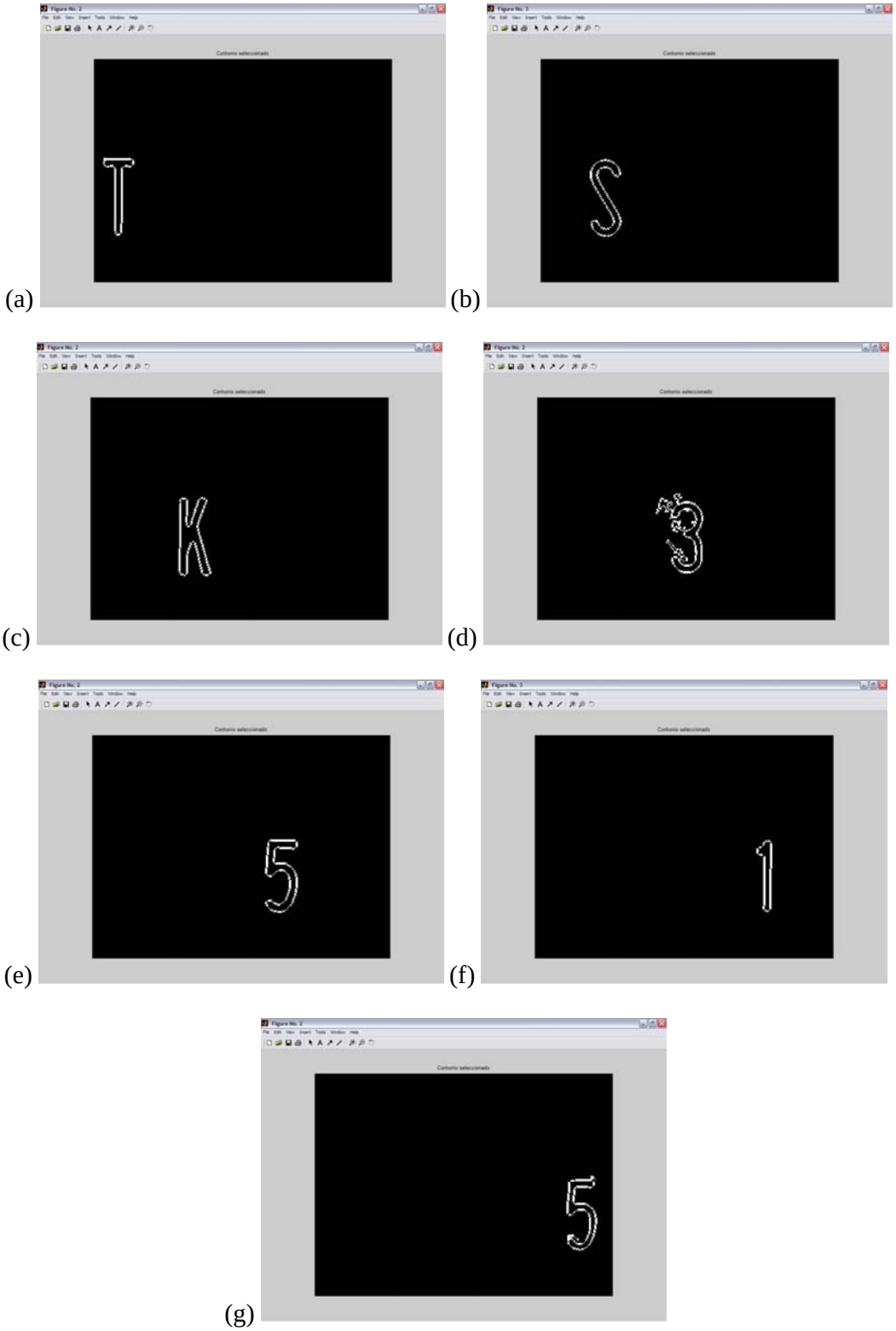

Figura 3.8. Extracción de contornos de: (a) Primera letra, (b) segunda letra, (c) tercera letra, (d) primer número, (e) segundo número, (f) tercer número, (g) cuarto número.

El comando *zeros* genera un arreglo de ceros (M), en éste caso se indica que el tamaño del arreglo es el mismo que el de la matriz BS.

El comando *size* despliega el tamaño del arreglo del argumento; dado que BS es una matriz bidimensional, *size* despliega el mismo número de columnas y de renglones y los guarda en las variables r y c respectivamente.

El ciclo *for* que sigue realiza un barrido de la matriz BS y al encontrar un 0, escribe 255 en la matriz M en esa posición; cuando BS es 1 binario, M conserva el 0. El barrido se realiza para que el contorno del objeto (representado por "unos" dentro de BS) se convierta en una referencia al valer 0 y tener distancia igual a cero con respecto a si mismo. Los demás puntos se inicializan a una distancia lo mayor posible.

```
%transformada de distancia
[ren,col]=size(M);
V1=zeros(1,6);
V2=zeros(1,6);
```

Con éstas declaraciones se inicializan dos vectores (V1 y V2) con seis posiciones y con valor inicial igual a 0.

```
for j=2: ren-1,
   for i=2: col-1,
      pixel1=(1.4142 + M((j-1),(i+1)));
      V1(1)=pixel1;
```

```
    pixel2=(1 + M((j-1),i));
    V1(2)=pixel2;
    pixel3=(1.4142 + M((j-1),(i-1)));
    V1(3)=pixel3;
    pixel4=(1+ M(j,(i-1)));
    V1(4)=pixel4;
    pixel5=(1.4142 + M((j+1),(i-1)));
    V1(5)=pixel5;
    pixel6=M(j,i);
    V1(6)=pixel6;
    minv=min(V1);
    M(j,i)= minv;
  end
end
```

El barrido iniciado con el comando *for* se lleva a cabo a partir del segundo renglón al penúltimo y de la segunda columna a la penúltima. En este barrido el vector V1 se va llenando con los valores que encuentra tras la comparación de un píxel pivote con sus píxeles vecinos. El operador de comparación se muestra en la Figura 3.9.

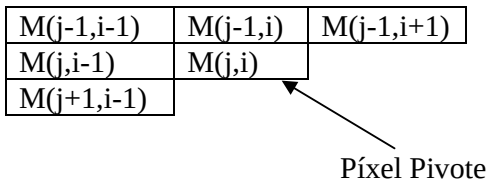

Figura 3.9. Operador de comparación.

El valor del píxel pivote se modifica tomando el valor mínimo de las casillas del vector sucesivamente. Los valores del píxel pivote solo se modifican drásticamente cuando encuentra al contorno. Como el barrido es de izquierda a derecha y de arriba hacia abajo, los píxeles que se modifican son los que se encuentran a la derecha y abajo del contorno. Para modificar el resto de los píxeles, es necesario un segundo barrido.

```
for j= ren-1: -1: 2,
   for i= col-1: -1: 2,
      pixel11=(1.4142 + M((j-1),(i+1)));
      V2(1)=pixel11;
      pixel12=(1 + M(j,(i+1)));
      V2(2)=pixel12;
      pixel13=(1.4142 + M((j+1),(i+1)));
      V2(3)=pixel13;
      pixel14=(1+ M((j+1),i));
      V2(4)=pixel14;
      pixel15=(1.4142 + M((j+1),(i-1)));
      V2(5)=pixel15;
      pixel16=M(j,i);
      V2(6)=pixel16;
      minva=min(V2);
      M(j,i)= minva;
   end
end
```

El segundo barrido efectuado es de derecha a izquierda y de abajo hacia arriba. Este barrido modifica los píxeles que se encuentran a la izquierda y arriba del contorno al encontrar el menor valor entre los píxeles vecinos. La Figura 3.10 muestra el operadot de comparación del segundo barrido.

		M(j-1,i+1)
	M(j,i)	M(j,i+1)
M(j+1,i-1)	M(j+1,i)	M(j+1,i+1)

Figura 3.10. Operador de comparación del segundo barrido.

```
matriz1=M;
objeto1= input('Nombre del objeto en la base de datos: ','s');
fprintf('Se ha terminado la transformada de distancia de ')
disp(objeto1)
disp('Si se quiere ver la transformada de distancia, llamar a la variable matriz1')
```

Estos comandos finales llevan a cabo la asignación de valores y nombre. Asignan la matriz M (que ya contiene la Transformada de Distancia del objeto) a una variable identificable por el usuario, y además le piden que de una etiqueta al objeto que ya se conoce.

Los subprogramas entrenamiento1, 2, 3 y 4 tienen las mismas características y siguen el mismo procedimiento, la diferencia radica en la imagen inicial a la que aplican el procesamiento y el nombre final de la variable que contiene a la Transformada.

El reconocimiento se lleva a cabo por medio de una función llamada reconocimiento.m

%prueba o comparacion final (reconocimiento.m)

function [prom]= reconocimiento(I)

procesamiento2(I)

seleccion

mapeo

reconocimiento es una función, por lo que requiere un argumento de entrada. El usuario debe ingresar el nombre de la imagen en la forma I=(‘nombre . formato’) y posteriormente escribir en la línea de comando: reconocimiento(I). Esto hará que se ejecuten 3 subprogramas.

%procesamiento de imagen y contorno del objeto

function [ ] = procesamiento2(I)

global B3;

%el programa lee la imagen

I1=imread(I);

I2=I1(:,:,1);

E=edge(I2,'canny');

E2=bwperim(E);

B2=bwmorph(E2, 'close');

B=bwfill(E2, 'holes');

```
B1=bwmorph(B, 'remove');
B3=bwmorph(B1, 'thicken');
%se muestran los contornos de las imagenes
figure,imshow(B3)
%title('Contornos de la Imagen')
```

Se puede observar que la imagen se procesa igual que las imágenes que forman parte de la base de datos, para evitar diferencias en el etiquetado e identificación de los objetos.

```
%extracción de contorno
global B3

BS=bwselect(B3, 8);
figure, imshow(BS)
title('Contorno seleccionado')
```

El reconocimiento utiliza la misma herramienta de Matlab que el entrenamiento para extraer el contorno de un objeto.

```
%mapeo o comparacion
global matriz1
global objeto1
global matriz2
global objeto2
```

```
global matriz3
global objeto3
global matriz4
global objeto4

%comparacion con el primer objeto
[r,c]=size(BS);
suma=0;
k=0;
for j=1: r,
   for i=1: c,
      elem=BS(j,i);
      if (elem==1)
        suma= suma + matriz1(j,i);
        k= k+1;
      end
  end
end

prom1=suma/k;
```

Este barrido recorre el arreglo del objeto que se quiere reconocer (BS) y la matriz con la Transformada de Distancia. Donde encuentra un píxel de contorno (1 binario) en BS, busca el valor de la Transformada en esa coordenada, posteriormente obtiene un

promedio de todos los valores encontrados, y a partir de éste toma las decisiones de reconocimiento.

```
fprintf('La distancia del objeto con respecto al primer objeto = ')
disp(prom1)
if(prom1==0)
   fprintf('El objeto es un(a) ')
   disp(objeto1)
end
```

Se lleva a cabo el mismo barrido para cotejar el contorno y los arreglos de las otras transformadas.

```
Vprom(1)=prom1;
Vprom(2)=prom2;
Vprom(3)=prom3;
Vprom(4)=prom4;

valormin= min(Vprom);

switch (valormin)
        case valormin==prom1
          fprintf('El objeto es un(a) ')
          disp(objeto1)
        case valormin==prom2
```

```
    fprintf('El objeto es un(a) ')
    disp(objeto2)
  case valormin==prom3
    fprintf('El objeto es un(a) ')
    disp(objeto3)
  otherwise
    fprintf('El objeto es un(a) ')
    disp(objeto4)
end
```

En teoría, una imagen se reconoce cuando el valor del promedio es igual a cero, sin embargo, eso implica que los objetos en todas las fotografías deben estar en la misma posición y ser exactamente del mismo tamaño. También implica que no exista ruido alguno que induzca variaciones en los píxeles. En la práctica es difícil encontrar fotografías de este tipo, y por ello fue necesario crear una base de datos con fotografías tomadas bajo las condiciones necesarias. Sin embargo, aún así los objetos dentro de las imágenes no son idénticos, por ello fue necesario comparar las distancias promedio del objeto con las cuatro transformadas, y asignar al objeto la etiqueta de la transformada que más se le parezca, es decir, con la que menor distancia tenga.

3.5 Descripción de la Base de Datos

El código del reconocimiento fue diseñado para soportar el conocimiento de 4 objetos al mismo tiempo. La base de datos del programa está conformada 4 imágenes temporales. Es importante mencionar que ésta base de datos puede ampliarse o sustituirse

de acuerdo a las necesidades del usuario, limitándose únicamente por las capacidades del sistema que lo ejecute.

Los objetos que se quieren reconocer deben aparecer en imágenes que los presenten con las siguientes características:

- Con la misma perspectiva (mismos ángulos) que las imágenes conocidas.
- Sin deformaciones.
- Sin inversión ni rotación.
- Con tamaños similares y sin escalamientos.
- Con colores similares o diferentes.

Las imágenes con las que se ha probado el programa han sido objetos tridimensionales (latas, botellas y libros) y objetos bidimensionales (placas), debido a las aplicaciones buscadas con este proyecto.

Las imágenes con las que se presentan los resultados del programa son placas de automóviles del Estado de Puebla. Todas son fotografías tomadas en la misma perspectiva, sin deformaciones, con tamaños similares. No fue posible retirar las placas de los automóviles, por lo que las letras y los números en ellas no son exactamente iguales.

Las características de las imágenes se pueden obtener con el comando *imfinfo* de Matlab, el cual arroja los siguientes datos de las imágenes utilizadas en las pruebas:

info =

Filename:'imagen.jpg'

FileModDate: '27-Nov-2003 19:28:44'

FileSize: 84512

Format: 'jpg'

FormatVersion: ''

Width: 400

Height: 300

BitDepth: 24

ColorType: 'truecolor'

FormatSignature: ''

Las imágenes que se utilizaron para probar el programa son fotografías de 400x300, imágenes de color verdadero o RGB y formato .JPG. El Anexo 1 muestra las imágenes utilizadas.