Apéndice. Programa Escrito en Matlab para Implementar el Algoritmo

A.1 Breve introducción a MATLAB

MATLAB es un lenguaje de computadora de alto nivel y un ambiente interactivo que permite llevar a cabo operaciones y tareas computacionalmente intensas más rápido que en otros lenguajes como C, C++ y Fortran. Entre estas tareas se encuentran el desarrollo de algoritmos, visualización de datos y cómputos numéricos. MATLAB puede ser usado para una amplia gama de aplicaciones incluyendo análisis de señales y de imágenes, comunicaciones, control, pruebas y mediciones, análisis de modelos financieros y biología computacional. Cuenta con herramientas agregadas llamadas *toolboxes* que son colecciones de funciones con propósitos especiales, que están disponibles por separado y que extienden la capacidad del ambiente MATLAB para resolver problemas específicos en áreas específicas.

El nombre de MATLAB es la abreviación de MATrix LABoratory (inglés para laboratorio de matrices) y surge justamente porque soporta operaciones de vectores y matrices que son fundamentales en la resolución de muchos problemas de ingeniería y ciencia. Lo que permite un rápido desarrollo y ejecución de código.

La ventana principal de MATLAB, llamada escritorio de MATLAB, luce como en la figura 1.6. En ésta es posible observar 5 subventanas: la ventana de comando (command window), el buscador en el espacio de trabajo (workspace),

el directorio activo (current directory), la historia de comandos (command history) y una o más ventanas de figura que se muestran sólo cuando el usuario desea desplegar una imagen o gráfica.

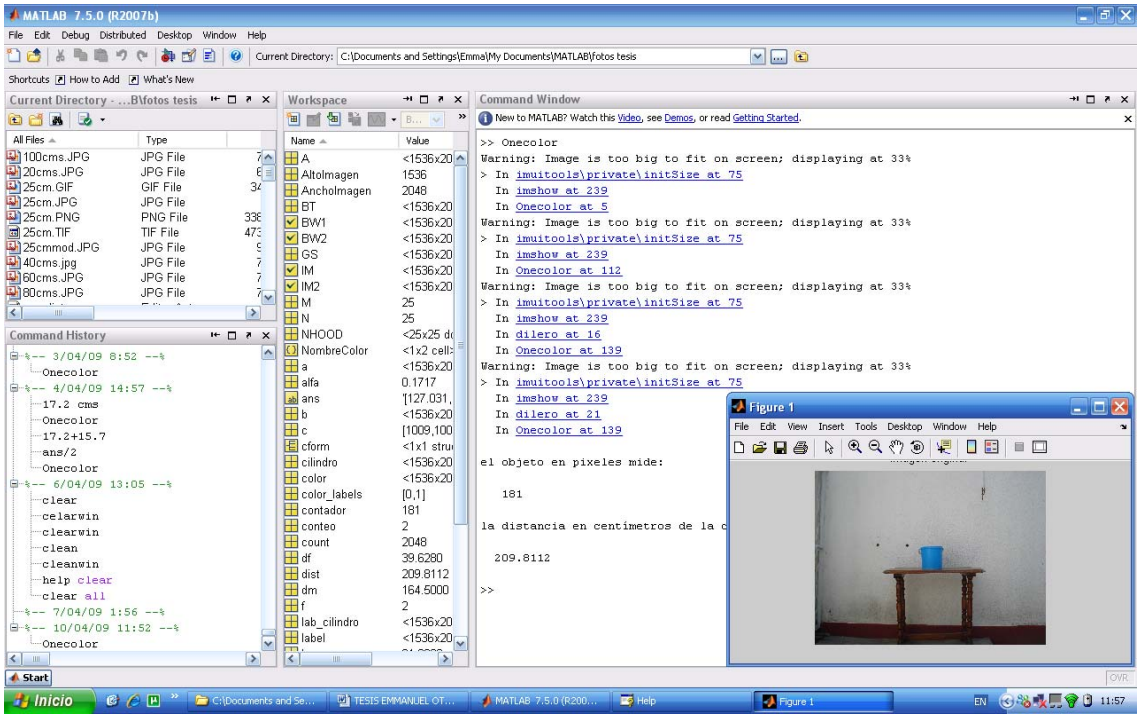

Figura 1.6 Escritorio de MATLAB

A.2 Programa escrito para la realización del algoritmo

El algoritmo consta de 5 funciones escritas que no son parte de MATLAB. El funcionamiento completo de las demás funciones ocupadas (y que son pertenecientes a MATLAB y al IPT) pueden ser consultadas en las ventanas de ayuda de MATLAB o tecleando *help* seguido del nombre de la función, dentro del entorno MATLAB

A.2.1 Función Onecolor.m

```
%Algoritmo para medición de distancias: Cuerpo del programa
%El archivo .m para esta función se llama Onecolor.m


A=imread('prueba1.jpg');
cilindro=imresize(A, [1536 2048]);%[404 539]);
figure
imshow(cilindro)
%figure
title('imagen original');
%info=imfinfo('60cms.jpg');

%Esta parte carga la imagen y la redimensiona para que tenga dimensiones
%equivalentes a una de 3 megapíxeles

AltoImagen=1536;     %info.Height;
AnchoImagen=2048;    %info.Width;

NombreColor = { 'cilindro','fondo'};
nColores = length(NombreColor);
size(nColores);
nColores;
muestra_region = false([AltoImagen AnchoImagen nColores]);

%En esta parte se inicializan las matrices donde se guardará la
%información de las muestras de los colores

%c=[265 265 273 273];%X para el color
%r=[194 206 206 194];%Y para el color
%p=[1 30 30 1];%X para el background
%q=[195 195 364 364];%Y para el background

c=[1009 1009 1039 1039];%X para el color
r=[745 791 791 745];%Y para el color
p=[1 30 30 1];%X para el background
q=[195 195 364 364];%Y para el background

% Selección de cada muestra de región
%Las partes con signo % para comentarios son las opciones que
%originalmente se usaron, para una foto de 539 x 404 píxeles
```

```
f = figure;
for conteo = 1:nColores
    if conteo==1
    set(f, 'name', ['Seleccione la mestra de la región para '
NombreColor{conteo}] );
    muestra_region(:,:,conteo) = roipoly(cilindro,c,r);
    else
        set(f, 'name', ['Seleccione la muestra de la región para '
NombreColor{conteo}] );
        muestra_region(:,:,conteo) = roipoly(cilindro,p,q);
    end

end

%En esta parte se automatiza la funcion de roipoly, ya que si no genera
%una ventana interactiva donde seleccionar el color del cilindro y luego
%el background, esto evita este paso y selecciona las áreas del cilindro
y back automáticamente.

close(f);


%for conteo = 1:nColores
 %   figure
  %  imshow(muestra_region(:,:,conteo))
   % title(['region de muestra para ' NombreColor{conteo}]);
   %end

%Muestra las regiones de color en caso de que se deseen observar para
%ver cómo está funcionando el programa gráficamente, no es una parte
%necesaria para el funcionamiento del programa, únicamente para
%comprobaciones de funcionamiento.

% Las siguientes instrucciones, convierten la imagen RGB del cilindro en
una imagen L*a*b.

cform = makecform('srgb2lab');
lab_cilindro = applycform(cilindro,cform);

%A continuación se calcula el promedio de los valores 'a' y 'b' para
%cada área extraída, las cuales sirven como marcadores de color en el
%espacio 'a*b'.

a = lab_cilindro(:,:,2);
b = lab_cilindro(:,:,3);
marcadores_color = repmat(0, [nColores, 2]);

for conteo = 1:nColores
  marcadores_color(conteo,1) = mean2(a(muestra_region(:,:,conteo)));
  marcadores_color(conteo,2) = mean2(b(muestra_region(:,:,conteo)));
end

%El promedio de color en las regiones de muestra en el espacio 'a*b' es:

for conteo = 1:nColores
    %disp
    ( sprintf('[%0.3f,%0.3f]', marcadores_color(conteo,1),
marcadores_color(conteo,2)) );
end
```

```
% Se crea un arreglo (vector) que contenga las etiquetas de los colores:
%     0 = fondo
%     1 = cilindro

color_labels = 0:(nColores-1);

%Se inicializan las matrices que se usan para la clasificación por medio
del vecino más cercano.

a = double(a);
b = double(b);
distance = repmat(0,[size(a), nColores]);



% En esta parte se realiza el procedimiento para la clasificación,
empezando por la aplicación del algoritmo del vecino más cercano.

for conteo = 1:nColores
  distance(:,:,conteo) = ( (a - marcadores_color(conteo,1)).^2 + ...
                       (b - marcadores_color(conteo,2)).^2 ).^0.5;
end

[value, label] = min(distance, [], 3);
label = color_labels(label);
clear value distance;
rgb_label = repmat(label, [1 1 3]);
segmented_images = repmat(uint8(0), [size(cilindro), nColores]);

%Teniendo las distancias mínimas se realiza la clasificación

for conteo = 1:nColores
  color = cilindro;
  color(rgb_label ~= color_labels(conteo)) = 0;
  segmented_images(:,:,:,conteo) = color;
end

%for conteo=1:nColores
%    figure
%    imshow(segmented_images(:,:,:,conteo));
%    title(['objetos con color de ' NombreColor{conteo}] );
%end

%Si se desea, al habilitar las 5 lineas anteriores, se muestran los
%objetos correspondientes a cada color segmentado

figure
imshow(segmented_images(:,:,:,1));
%figure

%Se convierte la imagen a escala de grises

GS=rgb2gray(segmented_images(:,:,:,1));
%imshow(GS);
%title('imagen gris');
%figure

%Se ajusta el contraste de la imagen

BT=imadjust(GS);
%imshow(BT);
```

```
%title('imagen gris mejorada');
%figure

%Se convierte de imagen en escala de grises a imagen binaria

BW1=im2bw(BT);
%imshow(BW1);
%title('imagen gris a blanco y negro');
%figure

%Se convierte de imagen en escala de grises a imagen binaria, previo
%ajuste de contraste

BW2=im2bw(imadjust(BT));
%imshow(BW2);
%title('imagen gris a blanco y negro mejorada');
%figure

%plot(BW2(202,:));
%title('histograma de la distribucion de pixeles');
%figure

%Si se desea ver el histograma del ancho en píxeles, se activan las tres
últimas líneas

%Las funciones que se quedan marcadas como comentarios con % están aquí
%porque sirven para comprobar la secuencia del funcionamiento del
%programa, gráficamente. Quedan como opciones.

dilero;

%La funcion dilero primero erosiona la imagen en blanco y negro para
%luego dilatarla y entonces regresar al tamaño original del objeto pero
%sin ruido, es decir, sin posibles pixeles que puedan interferir en el
%conteo de la función pixelweight.m.

%figure
%imshow(IM);
%title('imagen binaria final');

%figure
%plot(IM(202,:));
%title('distribución de pixeles blancos a lo ancho de la imagen');
%figure

Pixelweight;

%La function pixelweight mide el ancho en pixels del objeto que se aisló

formula;

%Con esta información, la función fórmula hace los cálculos necesarios
%para saber el tamaño virtual del objeto y la distancia que existe entre
%la cámara y el objeto.
```

A.2.2 Función dilero.m

```
%función dilero.m

%NHOOD=[1 1 1 1 1 1 1 1 1 1; 1 1 1 1 1 1 1 1 1 1;1 1 1 1 1 1 1 1 1 1;
%        1 1 1 1 1 1 1 1 1 1;1 1 1 1 1 1 1 1 1 1;1 1 1 1 1 1 1 1 1 1;
%        1 1 1 1 1 1 1 1 1 1;1 1 1 1 1 1 1 1 1 1;1 1 1 1 1 1 1 1 1 1;
%        1 1 1 1 1 1 1 1 1 1 ];

%NHOOD=[1 1 1;1 1 1;1 1 1];


%NHOOD=[1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1];

M=25;
N=25;
NHOOD = repmat(1,M,N);

%La función dilero usa una matriz de barrido, que es la matriz que
%erosiona y luego dilate la imagen. Esta puede usarse de las matrices
%predeterminadas que se encuentran en la parte superior ó, en su
%defecto, generarla usando la función repmat.

IM2=imerode(BW2,NHOOD);
figure
imshow(IM2);
title('imagen binaria erosionada');

IM=imdilate(IM2,NHOOD);
figure
imshow(IM);
title('imagen binaria dilatada');


%Las últimas ocho líneas muestran gráficamente los resultados de las
%erosión y la dilatación en la imagen binaria.
```

A.2.3 Función pixelweight.m

```
%Función pixelewight.m

contador=0;
for count=1:2048
    if (IM(768,count))       %Corresponde a la mitad de la imagen
        contador=contador+1;
    end
end


%La función píxelweight cuenta los pixeles blancos que ocupa el objeto a
%lo largo de la imagen
```

A.2.4 Función formula.m

```
%Función formula.m
disp('                                                ')
disp('el objeto en pixeles mide: ' ),
disp('                                                ')
disp(contador)


%alfa=0.6442;
alfa=0.171656728;

%La variable "alfa" es el factor de proporción que se utiliza para
%calcular el tamaño de la imagen virtual el cual se calcula
%multiplicando esta cantidad por el número de pixeles que ocupa el
%objeto a tratar, la alfa inhabilitada corresponde a alfa para imágenes
%de 539 x 404 píxeles.

n=contador;

%La variable "n", equivalente a contador, viene de la función
%pixelweight y lo que nos indica es el número de pixeles que ocupa el
%objeto en el plano de imagen. Esta cantidad se multiplicará por el
%factor de proporción "alfa" que fue calculado previamente, con el
%objetivo de obtener el tamaño del objeto en el plano de imagen
%equivalente a milímetros y no a %pixeles.

dm=47;

%La variable dm, "diametro", dice el diámetro del cilindro que se
%utilizó como objeto de prueba en el proyecto para calcular la distancia
%que existe de la cámara al mismo. Este dato puede ser cambiado cuando
%se usa un objeto de diferente tamaño, sin causar mal funcionamiento del
%programa

df=40.7602

%df=39.6280;
%La variable "df" se refiere a distancia focal. El valor de esta
%variable es un valor fijo que se calculó en el Capítulo 3 del
%presente trabajo. La distancia focal es parte esencial en la fórmula
%que permite calcular la distancia real del objeto con respecto a la
%lente de la cámara.

lp=alfa*n;

%La variable lp, "eleprima", es simplemente la multiplicación del factor
%de proporcion “alfa” y el número de pixeles que ocupa el objeto en el
%plano de imagen. Como se mencionó anteriormente, esta multiplicación
%nos dice el tamaño equivalente en milímetros del objeto prueba en el
%plano de imagen.

dist=dm*df/lp;

%la variable "dist" es el producto final de todos los cálculos
%realizados hasta el momento. Como se puede recordar en los Capítulos
%1 y 5 se explica la fórmula, el proceso para llegar a la misma y el por
%qué de su uso. Es la distancia real del objeto con respecto a la lente
%de la cámara.
```

```
disp('la distancia en centímetros de la cámara al objeto es de  ')
disp('                                                          ')
disp(dist)

%Las últimas tres líneas despliegan el resultado de la distancia al
%objeto desde la cámara.
```

A.3 Función triangulación.m

```
%Esta parte corresponde al código escrito para los experimentos de
%triangulación, no tiene conexión alguna con las funciones anteriores.
%Como se puede observar, este código consiste únicamente de la
%aplicación de las fórmulas de las leyes de los senos para el cálculo de
%distancias.

ALFA=19;
BETA=70;
BETAR=180-BETA;
LAMDA=180-ALFA-BETAR;
disAB=10;
%ALFA Y BETA siempre se miden con respecto a la misma referencia, ambos
%empiezan en el mismo cero y van en dirección contraria de las 5
%manecillas del reloj para medirse. Por este motivo al hacer la medición
%ALFA corresponde al ángulo interior del triángulo en el vértice A,
%mientras que BETA corresponde al ángulo exterior del triángulo ABC en
%el vértice B.

%BETAR es la corrección que se hace para que el ángulo coincida con el
%ángulo beta correspondiente al ángulo interior del triángulo, ya que la
%medición de BETA como tal es el ángulo exterior al vértice B.

AC=disAB*sind(BETAR)/sind(LAMDA);
BC=disAB*sind(ALFA)/sind(LAMDA);

RC=disAB*sind(BETAR)*sind(ALFA)/sind(LAMDA);
dist=disAB/(cotd(ALFA)+cotd(BETAR));

ALFA
BETA
disAB

AC
BC
dist
RC

%Las variables AC, BC y RC (que es del mismo valor de dist) equivalen a
%las distancias del triángulo de la Figura 5.2
```