ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Estudio e implementación de un Algoritmo para el seguimiento de color en video y la respectiva graficación de la ruta del objeto, usando la herramienta de adquisición y procesamiento de imágenes de Matlab"

REPORTE DE MATERIA DE GRADUACIÓN

Previo la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

José Antonio Baque Yoza

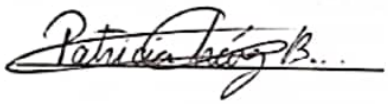
Guayaquil – Ecuador

Año 2009

AGRADECIMIENTO

Agradezco a mi Madre Ysidora por ser una mujer luchadora y cuyo ejemplo me da fuerzas para seguir adelante y a mis hermanos Javier, Rafael e Isabel quienes han estado junto a mi, brindándome su compañía, colaboración y atención en todo momento.

TRIBUNAL DE SUSTENTACIÓN



Ing. Patricia Chávez

Directora de Materia de Graduación

Ing. Rebeca Estrada

Delegada de la FIEC

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de éste Reporte de Materia de Graduación, me corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

RESUMEN

El presente artículo presenta la descripción de un programa desarrollado con Matlab el cual detecta un determinado color (vector RGB) en una secuencia de video que proviene de una cámara web o de un video en formato AVI almacenado en la memoria. De esta manera se puede hacer el seguimiento de un determinado objeto que se caracterice por su color.

El programa cuenta con una opción para que el usuario visualice la trayectoria del objeto en movimiento.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Si estuviéramos interesados en analizar a través de un video el **comportamiento del movimiento** de un objeto de color característico, como por ejemplo un satélite natural, satélite artificial o inclusive un avión, se podría desarrollar un programa **seguidor de movimiento por color**. Estos programas podrían ser utilizados en la NASA para dar seguimiento y control a satélites o estrellas, los cuales tienen normalmente colores muy particulares.

Desarrollaremos un programa en MATLAB que adquiera imágenes por medio de una **cámara web** y busque en tiempo real un color determinado por el usuario, a través de un vector de valores RGB. Adicionalmente, el programa graficará su trayectoria y posición.

El proyecto se lo realizará a nivel de software basado en el estudio del procesamiento de adquisición de imágenes considerando la resolución, número de bandas, trama por segundo entre otros parámetros.

Se realizará un estudio acerca de la forma como Matlab trabaja con la imágenes a través de la Matriz RGB, planos de la imagen, extracción de planos entre otras variables

La búsqueda de color dentro de una imagen, obtenida de una secuencia de video se centrará en el uso de un umbral de comparación entre la imagen de entrada y la matriz de color a buscar.

Para darle una mejor visualización al seguimiento del objeto, determinado por el color seleccionado, se generará una gráfica de la ruta que este objeto sigue.

CAPÍTULO 1

1. LA IMAGEN

1.1 Qué es una imagen digital?

Una imagen digital es una función que depende de 2 variables f(x,y), donde f es el brillo de un punto cuyas coordenadas espaciales son (x, y). A éste punto se le llama *píxel.*

1.2 Qué es el procesamiento digital de una imagen?

Es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

Cuando procesamos una imagen digital estamos convirtiendo la imagen en un número discreto de píxeles. El dispositivo que realiza éste proceso le asigna un número a cada píxel que especifica el brillo y el nivel de gris (gray-level). Una imagen digitalizada tiene 3 propiedades básicas: resolución, definición y número de planos.

1.3 Definición, Resolución y Número de Planos de una Imagen

La resolución de una imagen es el número de filas y columnas que forman los píxeles. Una imagen que tiene m filas y n columnas tiene una resolución m x n. Ésta imagen tiene n píxeles en su eje horizontal y m píxeles en su eje vertical.

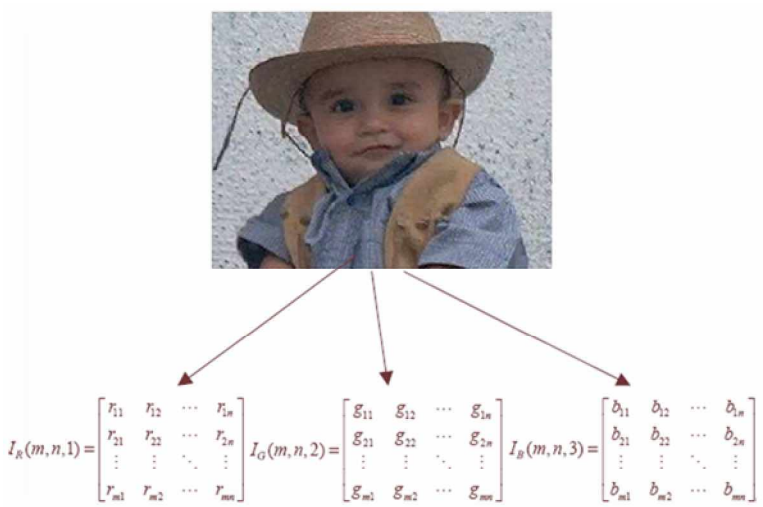


Figura 1.1 Representación Matricial (3 Matrices) de una imagen a color en Matlab.

La definición de una imagen, también llamada profundidad de píxel, indica el número de colores que se pueden ver en la imagen. La profundidad de modulación es el número de bits usados para codificar la intensidad de cada píxel. Un píxel puede tomar 2^n valores diferentes. Por ejemplo si n=8 el píxel puede tomar 256 valores en un rango entre negro (intensidad cero) y blanco (intensidad 255).

El número de planos es el número de arreglos de píxeles que componen la imagen. Una imagen con escala de grises (gray-level) está compuesta por un sólo plano, mientras una imagen de color verdadero (true-color) está formada por 3 planos que son: Rojo(R), Verde (G), Azul (B).

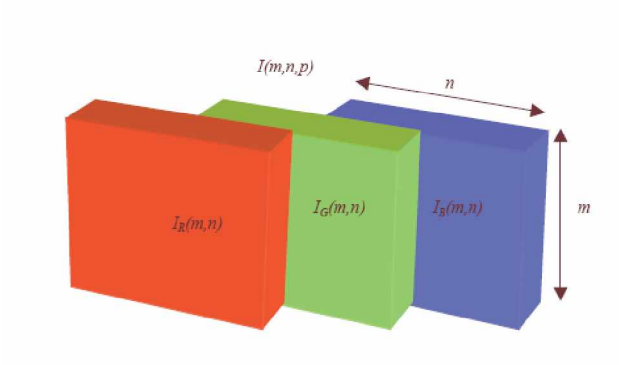


Figura 1.2 Planos Rojo, Verde y Azul que componen una imagen a color en Matlab.

CAPÍTULO 2

2. ADQUISICIÓN DE IMÁGENES

2.1 Funciones: imaqhwinfo, videoinput, getdata, imshow, aviread

La adquisición de imágenes para el procesamiento provienen de dos fuentes: cámara web y un archivo de video de formato AVI.

Para la adquisición de las imágenes con la cámara web se usó la herramienta de adquisición de imágenes, utilizando funciones tales como **imaqhwinfo** para determinar el hardware conectado y establecer sus propiedades.

La configuración de la cámara web se realiza con la función **videoinput**, configurada de tal manera que la adquisición sea de 50 tramas por disparo (trigger), el cual es activado por software. La adquisición empieza apenas inicia el video y tiene un intervalo de adquisición de trama de 1 (valor por defecto) para que la presentación del video sea considerada en tiempo real.

Se establecen 50 tramas por disparo debido a que al iniciar la cámara web las dos primeras imágenes son de valor 0 (pantalla negra) y las siguientes van poco a poco aclarando su nitidez. Por lo tanto, se dejan pasar las

primeras 50 tramas para estabilizar la imagen. La siguiente figura ilustra este fenómeno.

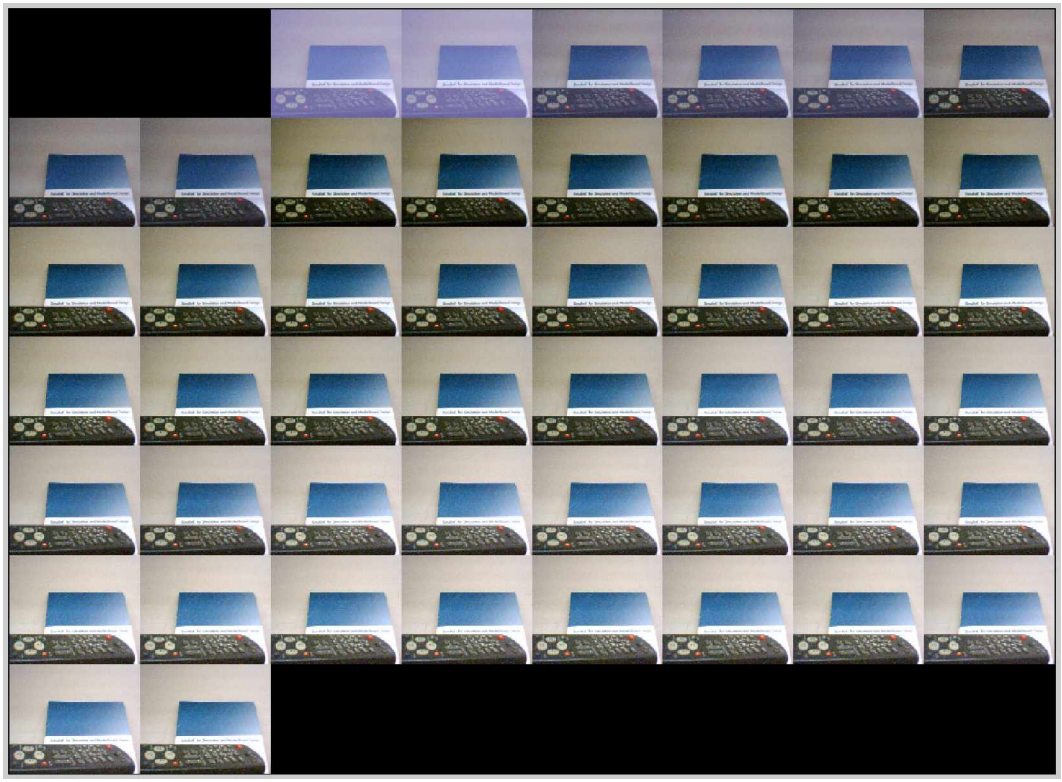


Figura 2.1 Adquisición de imágenes con cámara web.

Una vez establecida las propiedades de adquisición de la cámara web, se adquiere la imagen con la función **getdata,** la cual retorna por defecto una imagen en formato uint8 (entero sin signo de 8 bits). La función también permite establecer el formato de la imagen, por ejemplo double (número racional). Sin embargo, se conserva el formato uint8 para simplificar la presentación de la imagen con la función **imshow**.

Para la adquisición de la imagen de un video con formato AVI, se usa la función **aviread,** la cual retorna una estructura cuyo campo **cdata** es el que contiene las tramas del video.

CAPÍTULO 3

3. IDENTIFICACIÓN DEL COLOR

3.1 Ecuación "Diferencia" y Umbral

La identificación del color ser realiza mediante una comparación (resta) entre los planos RGB de la imagen de entrada con el valor RGB ingresado por el usuario.

$$Diferencia = [(PR_{img} - R_{us}) < umbral] \wedge [(PG_{img} - G_{us}) < umbral] \wedge [(PB_{img} - B_{us}) < umbral]$$

El **umbral** se determina de forma experimental, llegando a la conclusión que entre más semejanza tenga el color a buscar con el fondo mayor debe ser calibrado este umbral.

Esta ecuación retorna una matriz binaria del mismo tamaño que la imagen de entrada. Para precisar el color a buscar, se utilizan dos operaciones morfológicas para eliminar pequeñas áreas dentro de la imagen resultante.

Se usó dilación seguida por erosión usando la función **bwmorph** para luego realizar una erosión seguida por la dilación. Los resultados de estas operaciones se muestran en las siguientes figuras:

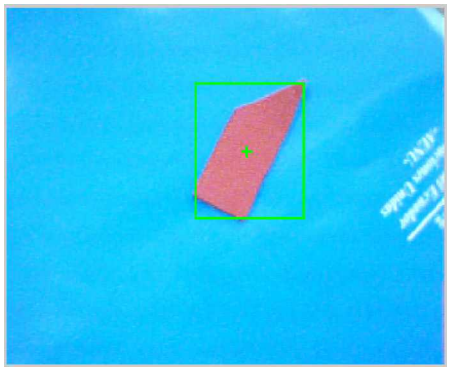


Figura 3.1. Imagen con el objeto a buscar ([255 138 139])

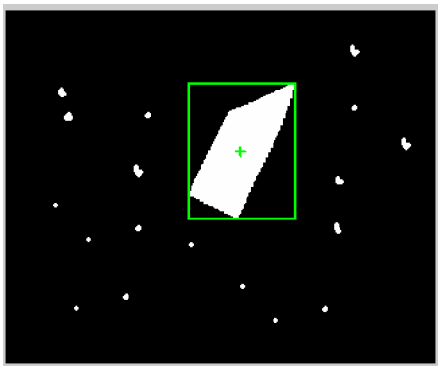


Figura 3.2. Imagen resultado de la Diferencia.

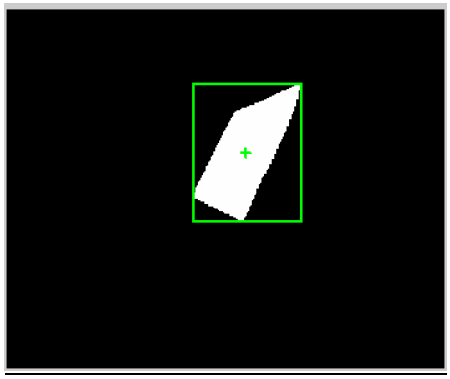


Figura 3.3. Imagen luego de realizar las operaciones morfológicas.

CAPÍTULO 4

4. ENCUADRE DEL OBJETO Y TRAZADO DE RUTA

4.1 Funciones: regionprops y bwlabel

Luego de haber obtenido la matriz de diferencias y de haber realizado las operaciones morfológicas, el siguiente paso es encontrar el área que representa al color buscado en la imagen. Para este fin se usa la función **regionprops**, la cual tiene como argumento de entrada una matriz de identificadores que se la obtenemos de la función **bwlabel** .

La función **regionprops,** en su forma básica, retorna tres parámetros: área, coordenada del centro y coordenada de un rectángulo. Todas estas son propiedades de los objetos (pixeles blancos) de la figura.

El rectángulo del objeto nos sirve para visualizar su movimiento, mientras que la coordenada del centro para graficar y almacenar su trayectoria. El área del objeto la usamos para eliminar los objetos menores a 200 pixeles.

La trayectoria se almacena en una matriz nula, concatenando cada valor de posición. La interfaz gráfica (GUI) del programa tiene una opción para almacenar la ruta recorrida usando la función **save** en un archivo .mat.

La siguiente figura muestra la GUI y la ruta del objeto:

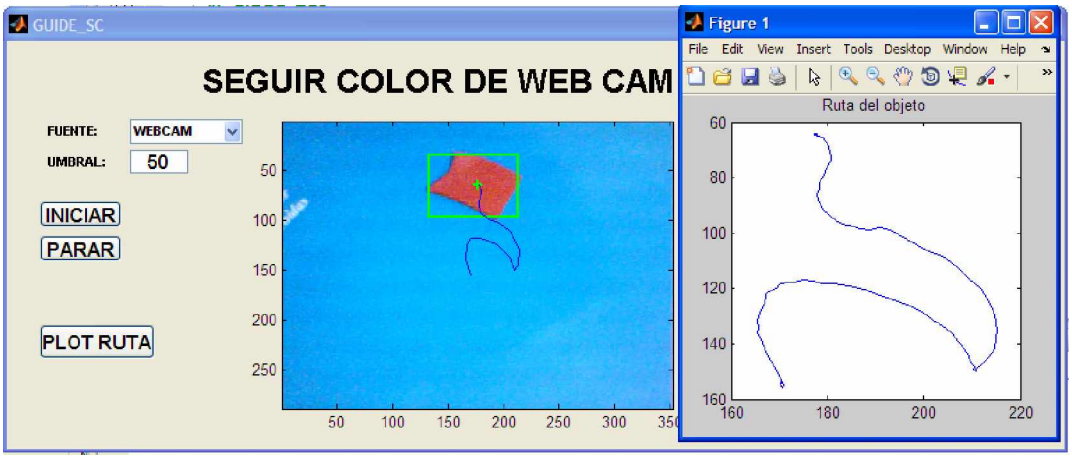


Figura 4.1 GUI de seguimiento de color y trazado de la ruta.

4.2 Diagrama de Flujo_Paralelos

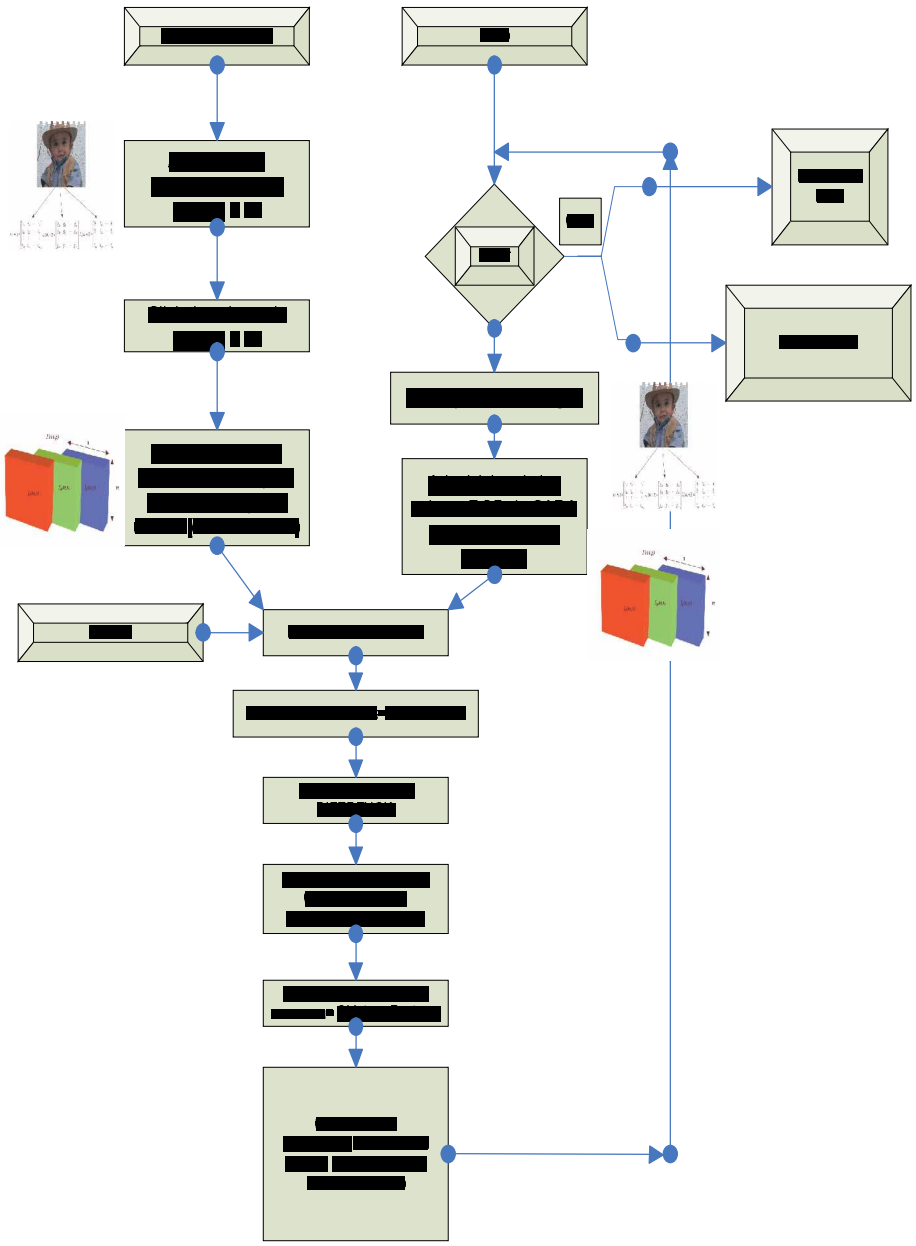


Figura 4.2 Diagrama de Flujo_Nivel Textual

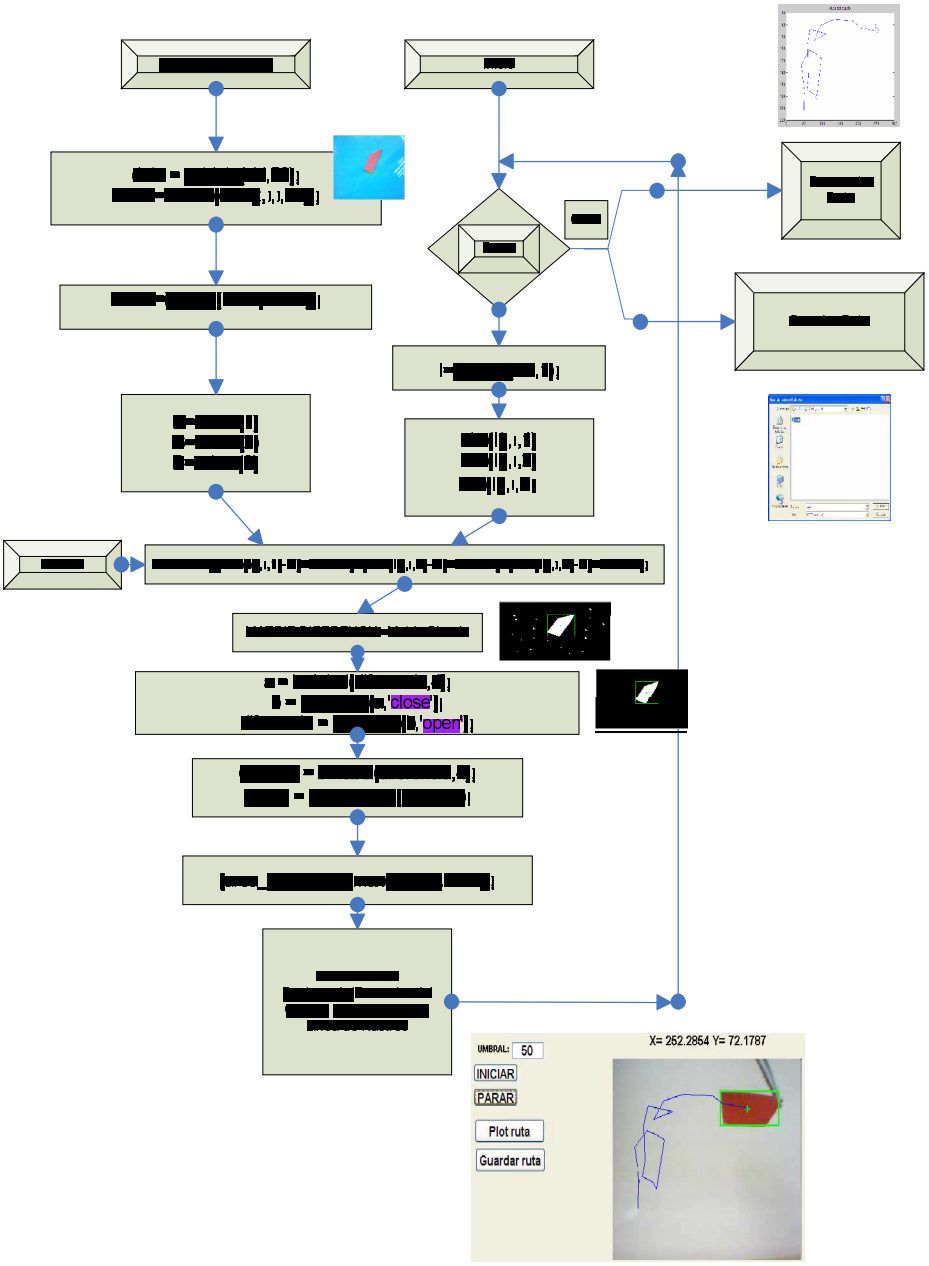


Figura 4.3 Diagrama de Flujo_Nivel Lineas de Código

CAPÍTULO 5

5. INTERFASE Y MANUAL DE USUARIO

5.1 Pasos: Seguir color de video

Figura 5.1 Interfaz

1. Seleccionar la fuente de video del POP- UP (menú) FUENTE

Figura 5.2 Selección FUENTE

2. Presionar el botón OBTENER DE IMAGEN para que aparezca un cursor. Hacer clic derecho sobre el color a buscar. Automáticamente se establecerá los valores de la matriz RGB en los cuadros de texto de la derecha:

Figura 5.3 Matriz RGB del color a buscar en la imagen

3. Establecer el umbral de comparación:

4. Hacer click en INICIAR para comenzar la búsqueda y trazado de ruta del objeto. El botón PARAR detiene la adquisición de imágenes:

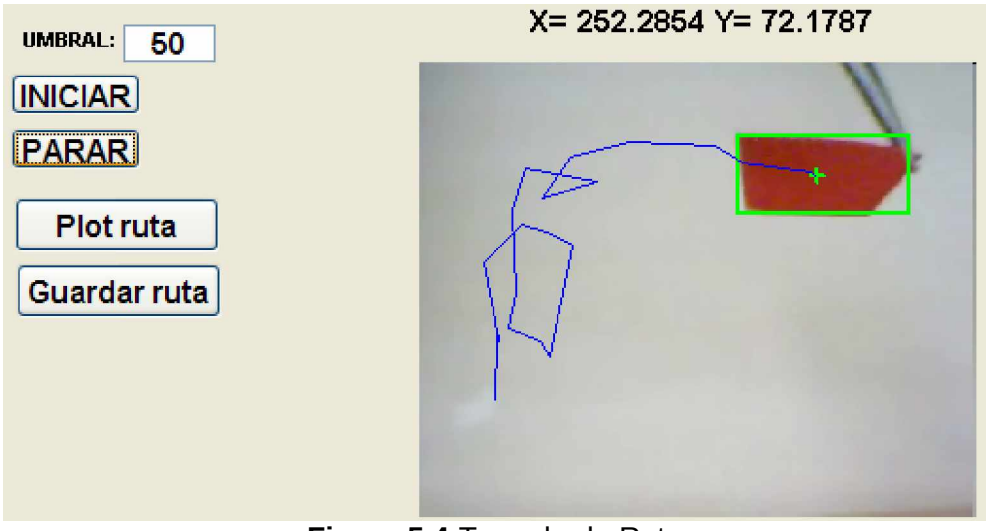


Figura 5.4 Trazado de Ruta

5. Una vez finalizada, el botón PLOT RUTA muestra la ruta en una nueva figura:

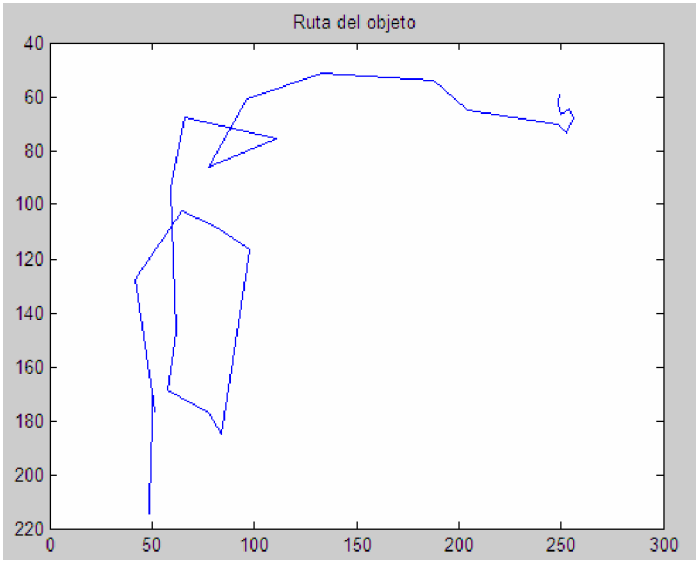


Figura 5.5 Trazado de Ruta con su respectiva escala

6. El botón GUARDAR RUTA salva la ruta en un archivo .mat:

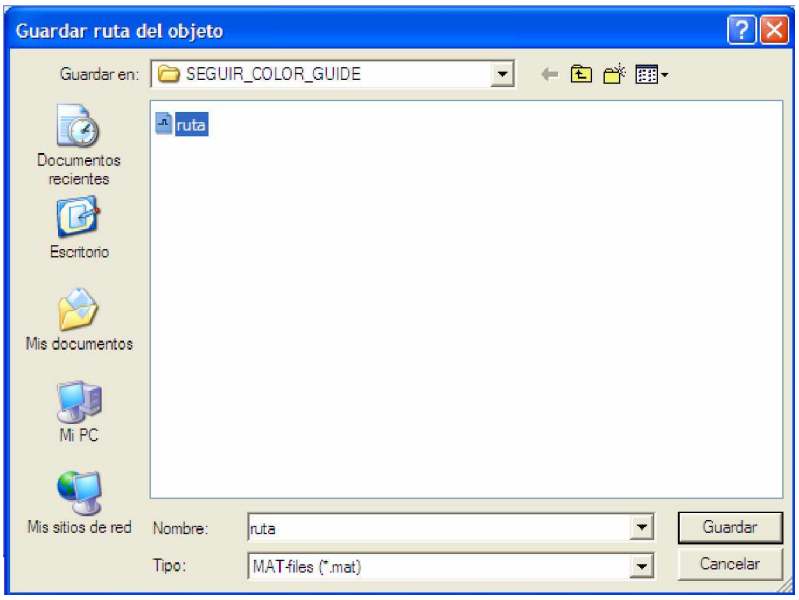


Figura 5.6 Guardar Ruta *.mat

CONCLUSIONES Y RECOMENDACIONES

Umbral 30->60. El umbral no debe ser muy alto, se recomienda entre 30 y 60, menor a 30 es probable que el programa no detecte el objeto con el color buscado y si es más de 60 es probable que se pierda la identidad del objeto debido a que hay un mayor nivel de tolerancia en los tonos del color a buscar y pueda que se rastree a un objeto distinto.

Entre más cercana está la matriz RGB del fondo(digamos la pared) a la matriz RGB del objeto, menor debe ser el valor del umbral, esto se debe al parecido y a que la diferencia de tonos será menor.

Si existen 2 o más objetos del mismo color, el programa sólo tomará en consideración para el rastreo al de máxima área.

El programa tiene un cierto grado de iteratividad debido a que el umbral se lo pude cambiar hasta encontrar el valor del umbral que optimize el rastreo.

La Iluminación es un factor muy importante para evitar ruido o mala interpretación del color a buscar. Además si se usa una webcam de mejor captura el programa trabajaría mejor ya que la nitidez de la imagen es también importante.

Las operaciones morfológicas abrir (open) y cerrar (close) reducen los espacios “blancos” que crea el ruido provocado por perturbaciones de la luz o por un umbral inadecuado.

Si se generan muchas trayectorias de forma casi aleatoria sólo cambie el valor del umbral a un valor inferior, éste le dará efectividad en la graficación de la trayectoria del color escogido por el usuario.

REFERENCIAS

[1] Cuevas Jimenez, Erik Valdemar; Zaldivar Navarro, Daniel, Visión por computador utilizando Matlab y la herramienta de procesamiento digital de imágenes

[2] Gonzáles, Wood, Eddins, Digital image proces

[3] Stephen Westland, Caterina Ripamonti, Computational Colour Science Using MATLAB

[4] Webinar de procesamiento de imágenes de mathworks

[5] http://proton.ucting.udg.mx/tutorial/vision/cursovision.pdf

[6] http://www.imageprocessingplace.com/

[7] Ing. Diego Barragán, Tutoriales, http://www.matpic.com

[8] http://www.matpic.com/VIDEOS_TUTORIALES.html

[9] http://www.youtube.com/user/diegokillemall

APÉNDICE

Apéndice A: GUIDE_SC.m

```
function varargout = GUIDE_SC(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GUIDE_SC_OpeningFcn, ...
    'gui_OutputFcn',  @GUIDE_SC_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%
_____
___
% --- CONDICIONES INICIALES DEL PROGRAMA
function GUIDE_SC_OpeningFcn(hObject, eventdata, handles, varargin)
% Mover la GU al centro
movegui(hObject,'center');
% String vacío para la ruta del video. Una vez que se elije el
video a leer
% este manejado (handles) se sobreescribe con la ruta del video
(Ej: C:\VIDEO\test.avi)
handles.ruta=' ';
% Deshabilitar los botones de "Trazar ruta" y "Guardar" hasta que
haya
% información disponible.
set(handles.trazar_ruta,'Enable','off');
set(handles.guardar,'Enable','off');
%
_____
___
% Choose default command line output for GUIDE_SC
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = GUIDE_SC_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
%
_____________________________________________________________________
___
%
_____________________________________________________________________
___
% BOTÓN QUE SELECCIONA LA IMAGEN DONDE DE ESCOGERÁ EL COLOR A
BUSCAR
function selec_fondo_Callback(hObject, eventdata, handles)
% Deshabilitar el botón durante el proceso
set(hObject,'Enable','off');
% Verificar la fuente de la imagen, si es la webcam o el video avi
sel_fuente=get(handles.video_f,'Value');
switch sel_fuente
    % ---------------------------------------------------------------
----
    case 1 %WEB CAM
        % Iniciar entrada de video
        vid = videoinput('winvideo', 1);
        % Intervalo de captuta de trama de la cadena de video
        vid.FrameGrabInterval = 1;
        set(vid,'TriggerRepeat',Inf);
        % Iniciar captura
        % _______Obtener el fondo_________
        disp('Obteniendo imagen de fondo')
        vid.FramesPerTrigger=50;
        start(vid);
        data = getdata(vid,50);
        fondo=double(data(:,:,:,50));
        imshow(uint8(fondo))
        stop(vid);
        clear vid
    case 2%VIDEO AVI
        [nombre ruta]=uigetfile('*.avi','Seleccionar video');
        % Si se presiona CANCELAR, retorna.
        if nombre == 0, return, end
        % Leer la secuencia de video AVI
        avi=aviread(fullfile(ruta,nombre));
        % Obtener las imágenes del video
        video = {avi.cdata};
        % Tomar la imagen número 10 (Este valor es empírico)
        fondo=video{10};
        % Almacenar la ruta y el nombre del archivo
        handles.ruta_s=fullfile(ruta,nombre);
        % Mostrar la imagen
        imshow(fondo)
end
```

```
% Usar IMPIXEL para tomar el valor RGB de la imagen con click
derecho.
colors=impixel(uint8(fondo));
% Retorna si no hay color selccionado
if isempty(colors)
    return
end
% Establecer la matriz RGB en los edit text.
set(handles.rojos,'String',colors(1));
set(handles.verdes,'String',colors(2));
set(handles.azules,'String',colors(3));
% Actualizar los valores de la GUI.
guidata(hObject,handles)
% Habilitar nuevamente el botón
set(hObject,'Enable','on');
%
_____________________________________________________________________
___
%
_____________________________________________________________________
___
% --- FUNCIÓN DEL BOTÓN "INICIAR"
function inicio_Callback(hObject, eventdata, handles)
% Habilitar los botones de "Trazar ruta" y "Guardar"
set(handles.trazar_ruta,'Enable','off');
set(handles.guardar,'Enable','off');
% Establecer el dato de usuario del botón PARAR en 0. Este botón
PARAR
% detiene la búsqueda de color
set(handles.parar_b,'UserData',0);
% Obtener el umbral de comparción de color
umbral=str2double(get(handles.umbral,'String'));
% Verificar que el valor del umbral es correcto, caso contrario
mostrar un
% mensaje de error y detener el programa
if umbral<0 || isempty(umbral) || isnan(umbral)
    errordlg('El umbral debe ser positivo y entero','Error de
umbral')
    return
end
% Obtener la matriz RGB de los edit-text
R=str2double(get(handles.rojos,'String'));
G=str2double(get(handles.verdes,'String'));
B=str2double(get(handles.azules,'String'));
% Verificar cual es la entrada de video (Webcam o Video AVI)
sel_fuente=get(handles.video_f,'Value');
% Matriz vacía que va a almacenar las coordenadas del objeto en
movimiento.
centros=[ ];
```

```
% El código que viene ejecuta la búsqueda en la imágenes que
provienes o
% bien de la webcam o bien del video AVI.
switch sel_fuente
    % ------------------------------------------------------------
----
    case 1 %WEB CAM seleccionada como fuente de video
        % Iniciar entrada de video
        vid = videoinput('winvideo', 1);
        % Repeticiones de captura infinita
        set(vid,'TriggerRepeat',Inf);
        % Intervalo de captuta de trama de la cadena de video
        vid.FrameGrabInterval = 1;
        % Iniciar captura
        start(vid);
        % Indicar el axes en donde se graficará la imagen con la
ruta
        axes(handles.axes1)
        try
            while 1 % Ciclo inf hasta presionar botón PARAR
                if get(handles.parar_b,'UserData')==1
                    break % Sentencia que termina con el ciclo
WHILE
                end
                % Extraer datos de la web cam
"getdata(vid,muestra)"
                I = getdata(vid,1);
                %Transformar imagen a formato de doble precisión
númerica
                I=double(I);
                % Cálculo de la referenica
                diferencia=(abs(I(:,:,1)-R)<umbral)&(abs(I(:,:,2)-
G)<umbral)&(abs(I(:,:,3)-B)<umbral);
                % Remover ruido (eliminar pequeños hoyos y rellena
aberturas)
                % 4 SIGNIFICA 4 objetos conectados
                a = bwlabel(diferencia,4);
                % Realiza cierre morfológico (dilación seguida por
la erosión)
                b = bwmorph(a,'close');
                % bwmorph: realiza operaciones morfológicas en
imágenes binarias
                % open->Realiza apertura morfológica (erosión
seguida por la dilación)
                diferencia = bwmorph(b,'open');
                % Transformar la imagen a UINT8 para mostrarla con
la
                % función IMSHOW
                imagen=uint8(I);
                image(imagen)
                % Etiquetar cada objeto encontrado
                etiqueta = bwlabel(diferencia,4);
```

```
                % Determinar propiedades de los objetos encontrados
como:
                % 'Area', 'Centroid', y 'caja'
                objeto = regionprops(etiqueta);
                % Contar el número de objetos encontrados
                N = size(objeto,1);
                if N < 1 || isempty(objeto) % Retorna si no hay
ningún objeto en la imagen
                    continue % Inicia el siguiente ciclo while.
                end
                % Eliminar los objetos con área menos a 200
(eliminar ruido)
                s=find([objeto.Area]<200);
                if ~isempty(s)
                    objeto(s)=[ ];
                end
                % Contar nuevamente el número de objetos
encontrados
                N=size(objeto,1);
                if N < 1 || isempty(objeto) % Retorna si no hay
ningún objeto en la imagen
                    continue
                end
                [area_maxi pam]=max([objeto.Area]);
                %for n=1:N
                    hold on
                    centroid = objeto(pam).Centroid;
                    C_X = round(centroid(1));
                    C_Y = round(centroid(2));
                    set(handles.coordenadas,'String',['X=
',num2str(C_X),' ','Y= ',num2str(C_Y)])
                    centros=[centros; C_X C_Y];

rectangle('Position',objeto(pam).BoundingBox,'EdgeColor','g','LineW
idth',2)

plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
                    plot(centros(1:5:end,1),centros(1:5:end,2));
                    hold off
                %end
                drawnow
            end
            %Parar adquisicón de video al terminar el ciclo While
            stop(vid);
        catch %Funciones que se ejecutan en caso de suceder un
error en la adquisición
            stop(vid);   %Parar la adquisición de imágenes
            delete vid  % Borrar el objeto de video  (ver imaqtool)
            clear vid    % Borrar el objeto de video del workspace
            errordlg('Sucedió un ERROR','Aviso') % Mostrar aviso de
error
        end
```

```
        % ------------------------------------------------------------
--------
    case 2 %VIDEO AVI seleccionado como fuente de video
        % Leer la ruta del video
        ruta=handles.ruta_s;
        % Si la "ruta" es un string vacío, muestra un mensaje y
retorna
        if strcmp(ruta,' ')
            msgbox('Seleccione un color a buscar del video
AVI','MENSAJE')
            return
        end
        % Matriz vacía que almacena las coordenadas de recorrido
del objeto
        centros=[ ];
        % Leer el video AVI
        avi=aviread(ruta);
        % Tomar las imágenes del video
        video = {avi.cdata};
        % El ciclo FOR procesará la totalidad de la imágenes del
video
        for cnt = 1:length(video)
            % Si se presiona el botón PARAR, rompe el lazo FOR
            if get(handles.parar_b,'UserData')==1
                break
            end
            % Presentar la imagen adquirida del video
            imagesc(video{cnt});
            % Desactivar la etiquetación de los ejes
            axis image off
            % Actualización de la imagen a presentar
            drawnow;
            % Convertir la imagen a double para operar.
            I=double(video{cnt});
            % Cálculo de la referenica
            diferencia=(abs(I(:,:,1)-R)<umbral)&(abs(I(:,:,2)-
G)<umbral)&(abs(I(:,:,3)-B)<umbral);
            % Remover ruido (eliminar pequeños hoyos y rellena
aberturas)
            % 4 SIGNIFICA 4 objetos conectados
            a = bwlabel(diferencia,4);
            % Realiza cierre morfológico (dilación seguida por la
erosión)
            b = bwmorph(a,'close');
            % bwmorph REALIZA OPERACIONES MORFOLÓGICAS EN IMÁGENES
BINARIAS
            % open->Realiza apertura morfológica (erosión seguida
por la dilación)
            diferencia = bwmorph(b,'open');
            % Etiquetar objetos encontrados
            etiqueta = bwlabel(diferencia,4);
```

```
% Determinar propiedades de los objetos encontrados como:
            % 'Area', 'Centroid', y 'caja'
            objeto = regionprops(etiqueta);%'Area', 'Centroid', y
'caja'
            % Contar el número de objetos encontrados
            N = size(objeto,1);
            if N < 1 || isempty(objeto) % Retorna si no hay ningún
objeto en la imagen
                continue
            end
            % Eliminar los objetos con área menos a 200 (eliminar
ruido)
            s=find([objeto.Area]<200);
            if ~isempty(s)
                objeto(s)=[ ];
            end
            % Contar el número de objetos encontrados
            N=size(objeto,1);
            if N < 1 || isempty(objeto) % Retorna si no hay ningún
objeto en la imagen
                continue
            end
            [area_maxi pam]=max([objeto.Area]);
%             for n=1:N
                hold on
                centroid = objeto(pam).Centroid;
                C_X = round(centroid(1));
                C_Y = round(centroid(2));
                set(handles.coordenadas,'String',['X=
',num2str(C_X),' ','Y= ',num2str(C_Y)])
                centros=[centros; C_X C_Y];

rectangle('Position',objeto(pam).BoundingBox,'EdgeColor','g','LineW
idth',2)

plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
                plot(centros(1:5:end,1),centros(1:5:end,2));
                hold off
%             end
            % Actualización de la imagen a presentar
            drawnow
        end
        disp('SALIO')
end
% Almacenar la ruta del objeto
handles.ruta=centros;
guidata(hObject, handles) %Actualizar datos de la GUI
% Habilitar botones "Trazar ruta" y "Guardar"
set(handles.trazar_ruta,'Enable','on');
set(handles.guardar,'Enable','on');
%
_____________________________________________________________________
___
```

```
%
_____
___
% --- TRAZAR LA RUTA
function trazar_ruta_Callback(hObject, eventdata, handles)
% Mostrar una nueva figura
figure(1)
% Llamar los datos de la ruta
ruta=handles.ruta;
% Plotear la ruta
plot(ruta(1:5:end,1),ruta(1:5:end,2))
title('Ruta del objeto')
axis ij
%
_____
___
%
_____
___
% --- FUNCIÓN PARA GUARDAR LA RUTA EN UN ARCHIVO .MAT.
function guardar_Callback(hObject, eventdata, handles)
% GUI de usuario para colocar el nombre del archivo
[nombre ruta]=uiputfile('*.mat','Guardar ruta del objeto');
%Retorna si se presiona cancelar
if nombre==0, return, end
% Guardar la ruta con la función SAVE
ruta=handles.ruta;
save(nombre,'ruta')
%
_____
___
%
_____
___
% --- FUNCIÓN DEL BOTÓN PARAR (DETIENE LA ADQ DE IMÁGENES)
function parar_b_Callback(hObject, eventdata, handles)
set(handles.parar_b,'UserData',1);
guidata(hObject, handles);
%
_____
___
%
_____
___

function video_f_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function video_f_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function umbral_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function umbral_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function rojos_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function rojos_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function verdes_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function verdes_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function azules_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all
properties.
function azules_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Apéndice B: GUIDE_SC.fig

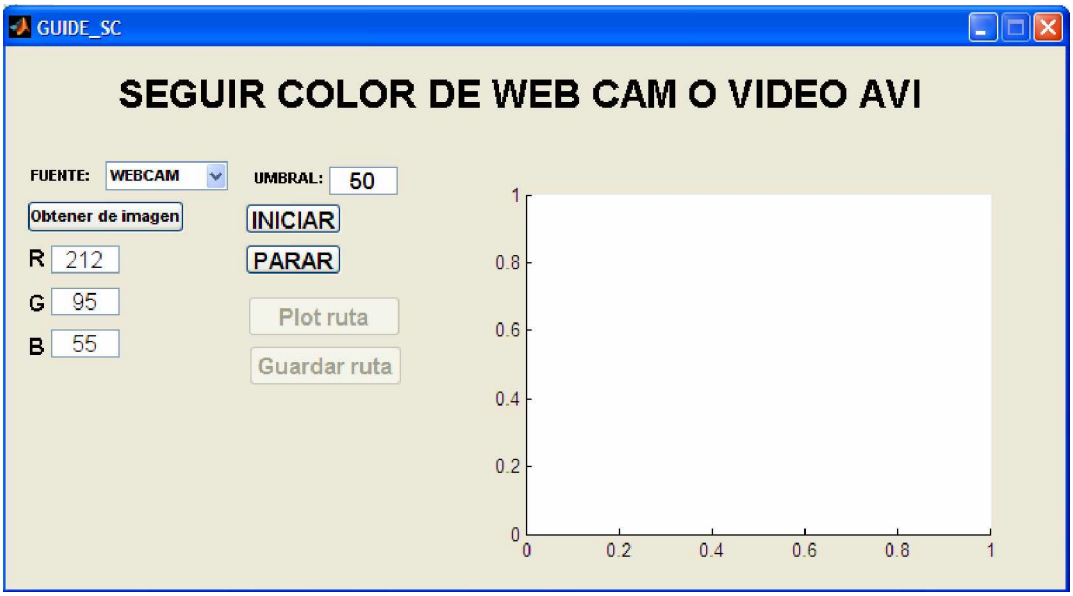


Figura Apéndice.1 GUIDE_SC.fig