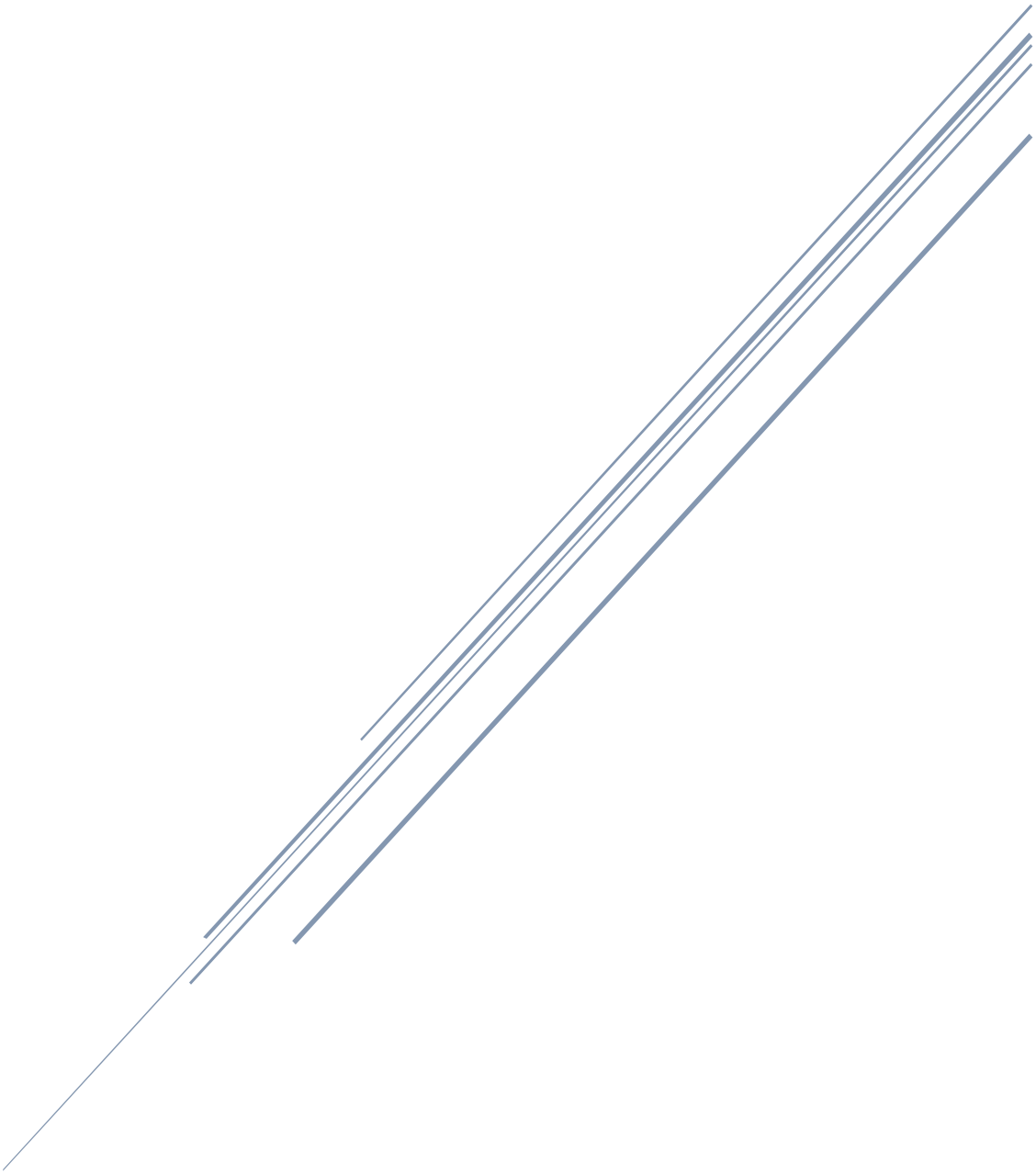
PRÁCTICA 3 – SEGMENTACIÓN DE CALZADA EN IMAGEN AÉREA

Sistemas de Percepción – 4 GIERM



Pablo León Barriga

2021/2022

Introducción

El programa se compone de un código principal, “MAIN_pr3.m”, y de una seria de funciones que serán llamadas desde ese código principal o desde otras funciones. Se pasa a explicar directamente la versión avanzada, pues que se trata de implementar la detección automática de los errores en la detección de los límites de la vía de interés.

Archivo MAIN_pr3.m

Se comienza inicializando una serie de variables que serán útiles para crear los bucles for que recorran las imágenes, y se arranca el subplot en el que se irán mostrando las imágenes una vez sean procesadas.

```
% Practica 3 - Sistemas de Percepcion - Pablo Leon Barriga
clc;clear;close all;
fprintf("Practica 3 - Sistemas de Percepcion, Pablo Leon Barriga\n");
fprintf("El programa se parara cada vez que se detecte un error, para continuar basta
con pulsar cualquier tecla.\n");
fprintf("Si se quiere ralentizar la muestra de imagenes, aumentar el valor de
""pausa"" (valor en segundos), en la linea 6.\n");
pausa = 0.00000001;

%% Inicializacion de variables/graficas

% Obtener tamano de la imagen a tratar
imagenLeer = sprintf('secuenciaBicicleta/000260.jpg');
fOriginal = imread(imagenLeer);
fResized = imresize(fOriginal, 0.30);
[nFilas, nColumnas, ~] = size(fOriginal);
[nFilasResized, nColumnasResized, ~] = size(fResized);

% Obtener una primera imagen tratada para tener una imagen binaria.
fTratada = pr3_Tratado_Imagen(260);

% Inicializar graficos
% Crear las graficas, para evitar hacer imshow una y otra vez.
figure('units','normalized','outerposition',[0 0 1 1])  % Pantalla completa
set(gcf,'color','white');       % Figura con fondo blanco
subplot(2,2,1);p1 = imshow(fResized, []);
subplot(2,2,2);p2 = imshow(fTratada, []);title('Lineas de interes, sin corregir');
subplot(2,2,3);p3 = imshow(fTratada, []);title('Lineas de interes, corregidas');
subplot(2,2,4);p4 = imshow(fResized, []);title('Segmentacion de la calzada');
pause(0.0001);  % Este pause sirve para darle tiempo a la ventana a crearse

% Contador de fallos
contador_fallos = 0;

%% Limites para la discrepancia entre valores de rho y theta
diag = (nFilas^2+nColumnas^2)^0.5;
limRho = 0.01*diag; % 1% de la diagonal
limTheta = 7.5*pi/180;   % 15 grados

rhoCorrecta = [0 0];
thetaCorrecta = [0 0];

primerFrame = 260;
ultimoFrame = 790;
```

Una vez inicializado las variables, se entra en un bucle for en el que se recorre una a una cada imagen del vídeo, son procesadas para obtener los límites de la calzada y se muestran en el

subplot. Se comienza llamando a la función "**pr3_Tratado_Imagen()**", función que devuelve la imagen tratada en binario, tras haber aplicado una segmentación de la imagen en el espacio HSV. A continuación se llama a la función "**pr3_Hough_Rho_Theta()**", que haciendo uso de la transformada de Hough, devuelve vectores con aquellos valores de *rho* y *theta* obtenidos de la matriz de votos de la transformada, así como la longitud de esos vectores.

```
%% Programa principal - recorre cada fotograma y opera
for i = primerFrame:ultimoFrame

    % Obtener la imagen tratada
    [fTratada, fResized] = pr3_Tratado_Imagen(i);

    % Calcular las rectas principales utilizando la transformada de Hough
    [nPicos,rhoHough,thetaHough] = pr3_Hough_RhoTheta(fTratada);
```

El siguiente paso es el de "corregir" los valores obtenidos mediante la transformada de Hough. Se parte de la suposición de que en la primera iteración se obtuvieron un par de rectas válidas para el desarrollo de la práctica. En la función "**pr3_Correccion_Rho_Theta()**" se comparan los dos valores más altos obtenidos en la iteración actual con la transformada de Hough con los de la iteración anterior, y se devuelven los vectores "corregidos", es decir, con las dos líneas identificadas.

A continuación se obtiene una plantilla de la carretera, aprovechando que se tienen los valores corregidos de las rectas de la calzada mediante la función "**pr3_Obtener_Plantillas()**". Se calculan las intersecciones de ambos pares de rectas (las "originales" y las "corregidas") a través de la función "**pr3_Calculo_Intersecciones()**".

```
% Corregir los valores de rho y theta. Se comprueba la correspondencia
    % entre las dos thetas y rho obtenidas en esta iteracion con las de la
    % iteracion anterior.
    [rhoCorrecta,thetaCorrecta, fallo] =
pr3_Correccion_Rho_Theta(rhoHough,thetaHough, i, rhoCorrecta, thetaCorrecta, limRho,
limTheta, nPicos,primerFrame);

    % Una vez se tienen los valores correctos de rho y theta (es decir,
    % manteniendo el orden que se establecion en la primera iteracion),
    % calcular la mascara que se corresponde con la segmentacion de la
    % calzada.
    [fMascara] = pr3_Obtener_Plantillas(nFilasResized,nColumnasResized,
thetaCorrecta, rhoCorrecta);

    % Calcular la interseccion de cada recta con los bordes de la imagen.
    %   - interseccionErronea: se calcula con los dos primeros valores que
    %       devuelve la funcion pr3_Hough_RhoTheta.
    %   - interseccionCorrecta: se calcula con los valores ya corregidos
    %       de Rho y Theta en la funcion pr3_Correccion_Rho_Theta
    interseccionErronea = pr3_Calculo_Intersecciones(rhoHough,thetaHough, nPicos,
nFilas, nColumnas);
    interseccionCorrecta = pr3_Calculo_Intersecciones(rhoCorrecta,thetaCorrecta,
nPicos, nFilas, nColumnas);
```

Por último en cada iteración se actualiza el subplot creado inicialmente, y en caso de fallo se informa al usuario, parando la reproducción de imágenes hasta que se pulse una tecla.

```
% Mostrar imagenes deseadas
```

```
    % 1 - Imagen de tamano reducido
    subplot(2,2,1);
    set(p1, 'CData', fResized);
    title(sprintf("Fotograma actual: 000%d.jpg", i));

    % 2 - Dos primeras lineas de Hough
    subplot(2,2,2);
    pr3_DibujaLineas(fTratada,interseccionErronea, nFilas, nColumnas, 0, i, p2,
primerFrame);

    % 3 - Lineas corregidas
    subplot(2,2,3);
    pr3_DibujaLineas(fTratada,interseccionCorrecta, nFilas, nColumnas, 1, i, p3,
primerFrame);

    % 4 - Imagen con la calzada segmentada
    subplot(2,2,4);
    set(p4, 'CData', uint8(fMascara).*fResized);

    % Informar de los fallos
    if fallo == 1
        contador_fallos = contador_fallos + 1;  % Aumentar el contador de fallos
        fprintf("Se produjo un fallo en la imagen secuenciaBicicleta/000%d.jpg.
Contador de fallos: %d. Pulse cualquier tecla para continuar.\n", i,
contador_fallos);
        pause();    % Parar el programa para dar tiempo a visualizar el error.
    end

    % Pausa para dar tiempo a actualizar cada subplot. Si se quiere
    % ralentizar la muestra de imagenes basta con aumentar el valor.
    pause(pausa);

end
```

Una vez analizadas todas las imágenes, se muestra por pantalla las estadísticas de fallos y aciertos.

```
%% Estadisticas

% Numero de fotogramas
nFrames = ultimoFrame - primerFrame;

% Porcentaje de fallos
error_relativo = contador_fallos / nFrames * 100;

% Mostrar por terminal
fprintf("Se produjeron %d fallos de un total de %d imagenes. Porcentaje de aciertos:
%.2f%%. Porcentaje de fallos: %.2f%%\n", contador_fallos, nFrames, 100 -
error_relativo, error_relativo);
```

Funciones

pr3_Tratado_Imagen.m

Función encargada de leer y tratar la imagen de manera que se obtenga una imagen binaria con especial atención en las líneas de la carretera. Al inicio del código se encuentran los umbrales de cada canal del espacio de colores HSV para binarizar la imagen. Se comienza leyendo la imagen que corresponde según el índice **i** recibido como argumento, y modificando su tamaño a un 30% del original. Una vez reducido, se binariza la imagen con los umbrales obtenido mediante la herramienta colorThresholder de MATLAB, y se realiza una operación de cerrado para eliminar algo del ruido generado con la binarización.

```
function [fTratada, fResized] = pr3_Tratado_Imagen(i)
%f1_Tratado_Imagen Funcion que lee y trata la imagen para obtener las
%lineas principales mediante la transformada de Hough

% Umbrales obtenidos mediante la herramiento colorThresholder
% umbralesHSV_nofallos = [ 0.187, 0.111, 0.590, 0.215, 0.839, 0.550];
umbralesHSV_fallos = [ 0.157, 0.111, 0.530, 0.215, 0.839, 0.650];
umbralesHSV = umbralesHSV_fallos;

% Leer la imagen, disminuir al 30% del tamano
imagenLeer = sprintf('secuenciaBicicleta/000%d.jpg', i);
fOriginal = imread(imagenLeer);
fResized = imresize(fOriginal, 0.30);

% Pasar a espacio de colores HSV
fHSV = rgb2hsv(fResized);

%Separar por canales
fH = fHSV( :,:,1); % canal hue
fS = fHSV( :,:,2); % canal saturation
fV = fHSV( :,:,3); % canal value

% El siguiente if comprueba que no se trate del caso en que el umbral
% minimo es mayor que el umbral maximo en el campo Hue.
if (umbralesHSV(1) < umbralesHSV(2))
    fTratada =  (fH < umbralesHSV(1) | fH > umbralesHSV(2)) &...
        (fS < umbralesHSV(3) & fS > umbralesHSV(4)) &...
        (fV < umbralesHSV(5) & fV > umbralesHSV(6));
else
    fTratada =  (fH < umbralesHSV(1) & fH > umbralesHSV(2)) &...
        (fS < umbralesHSV(3) & fS > umbralesHSV(4)) &...
        (fV < umbralesHSV(5) & fV > umbralesHSV(6));
end

 % Crear una plantilla en forma de diamante de radio 3
elem = strel('diamond', 3);
 % Realizar operaciones morfologicas para obtener el resultado deseado
fTratada = imclose(fTratada, elem);
end
```

pr3_Rho_HoughTheta.m

Función que recibe la imagen tratada obtenida de la función anterior, y obtiene mediante la transformada de Hough los vectores que contienen la información de las líneas "principales" de la imagen.

Se comienza calculando la transformada mediante la función **hough**. Se calculan los picos de la matriz de votos con la función **houghpeaks**, indicando un umbral de intensidad del 30% del pico máximos, y un entorno de vecindada de 31 valores, de manera que se evite encontrar picos cercanos entre ellos. Por último se crean los vectores **rhoHough** y **thetaHough** que serán devueltos.

```
function [nPicos,rhoHough,thetaHough] = pr3_Hough_RhoTheta(fTratada)
%f1_Hough_RhoTheta Funcion que obtiene los valores de rho y theta de las
%rectas principales de la imagen

% Aplicar la transformada de Hough para obtener las rectas principales
% de la imagen.
[H, tablaTheta, tablaRho] = hough(fTratada, 'Theta', -90:0.2:89);
Picos = houghpeaks(H, 20, 'Threshold', 0.3*max(H(:)),'nHoodSize', [31,31]);
nPicos = length(Picos(:,1));

% Obtener los valores en coordenadas polares de esas rectas
for k = 1:nPicos
    rhoHough(k) = tablaRho(Picos(k,1));
    thetaHough(k) = tablaTheta(Picos(k,2)) * pi / 180;
end

end
```

pr3_Correccion_Rho_theta.m

Función que recibe los valores corregidos del frame anterior (o los crea si se trata de la primera iteración del programa), y compara con ellos los valores obtenidos en la iteración actual, ordenando los parámetros de manera que la primera posición del vector se corresponda siempre con la recta que fue definida en la primera iteración, y que la segunda posición se corresponda con la segunda de la primera iteración.

Se comienza inicializando las variables **rho1**, **rho2**, **theta1** y **theta2**, que cogen los valores de la iteración anterior, y **fallo**, que indica si se ha producido un error. Se comprueba si se trata del primer fotograma, en cuyo caso se toman como bueno los dos primeros valores obtenidos en la función anterior. En caso contrario, se procede a comprobar la correspondencia entre las líneas detectadas en el fotograma actual frente a las obtenidas como buenas en el anterior.

```
function [rhoCorrecta,thetaCorrecta, fallo] =
pr3_Correccion_Rho_Theta(rhoHough,thetaHough, i, rhoCorrecta, thetaCorrecta, limRho,
limTheta, nPicos,primerFrame)
%f1_Correccion_Rho_Theta Funcion que compara y corrige los valores de rho y
%theta

% Inicializar valores
rho1 = rhoCorrecta(1);
rho2 = rhoCorrecta(2);
theta1 = thetaCorrecta(1);
theta2 = thetaCorrecta(2);



fallo = 0;
```

```
% Solo si nos encontramos en la primera iteracion, coger los dos
% valores de las rectas como validos (es importante comprobar que se
% parte de un par de rectas correctas).
if i == primerFrame
    rho1 = rhoHough(1); theta1 = thetaHough(1);
    rho2 = rhoHough(2); theta2 = thetaHough(2);
    rhoCorrecta = [rho1 rho2];
    thetaCorrecta = [theta1 theta2];
```

Pueden darse una serie de situaciones, explicadas a continuación.

1. Primera recta actual se corresponde con la primera recta anterior.
    a. Segunda recta actual se corresponde con la segunda anterior.
        i. No se considera como fallo.
    b. Segunda recta actual no se corresponde con la segunda anterior.
        i. Se considera fallo, se pasa a buscar en el resto de las rectas con aquella que coincida con la segunda recta del fotograma anterior.

```
else
    % Para el resto de fotogramas, se compara el valor obtenido para
    % cada rho y theta con los de la iteracion anterior, teniendo en
    % cuenta que puede haber cierto margen, definido en las variables
    % limTheta y limRho.



    if (abs(rhoHough(1)) < abs(rho1) + limRho && abs(rhoHough(1)) > abs(rho1) -
limRho) &&...
            (abs(thetaHough(1)) < abs(theta1) + limTheta && abs(thetaHough(1)) >
abs(theta1) - limTheta)
        % Si la primera recta se corresponde con la primera recta en el
        % apartado anterior.
        %disp('El primero se corresponde con el primero');
        rho1 = rhoHough(1); theta1 = thetaHough(1);

        % Comprobar ahora si la segunda recta se corresponde con la
        % segunda del frame anterior.
        if (abs(rhoHough(2)) < abs(rho2) + limRho && abs(rhoHough(2)) > abs(rho2) -
limRho) &&...
                (abs(thetaHough(2)) < abs(theta2) + limTheta && abs(thetaHough(2)) >
abs(theta2) - limTheta)
            %disp('El segundo se corresponde con el segundo');
            rho2 = rhoHough(2); theta2 = thetaHough(2);
        else
            % En caso de que la segunda linea no se corresponda, se
            % debe buscar en el resto de rectas encontradas con la
            % transformada de Hough.
            fallo = 1;
            for j = 3:nPicos
                if (abs(rhoHough(j)) < abs(rho2) + limRho && abs(rhoHough(j)) >
abs(rho2) - limRho) &&...
                        (abs(thetaHough(j)) < abs(theta2) + limTheta &&
abs(thetaHough(j)) > abs(theta2) - limTheta)
%                     fprintf("El segundo se corresponde con j: %d\n", j)
                    rho2 = rhoHough(j); theta2 = thetaHough(j);
                    %                         pause;
                    break;
                end
            end
            %disp('Error');
        end
```

2. Segunda recta actual se corresponde con la primera recta anterior.
    a. Primera recta actual se corresponde con la segunda anterior.
        i. No se considera fallo.
    b. Primera recta actual no se corresponde con la segunda anterior.
        i. Se considera fallo, se pasa a buscar en el resto de rectas actuales aquella que coincida con la segunda recta del fotograma anterior.

```
        % En caso de que la recta 1 no se corresponda con la recta 1
        % del frame previo, comprobar si la recta 2 obtenida se
        % corresponde con la primera del frame anterior.
    elseif (abs(rhoHough(2)) < abs(rho1) + limRho) && abs(rhoHough(2)) > abs(rho1) -
limRho &&...
            (abs(thetaHough(2)) < abs(theta1) + limTheta && abs(thetaHough(2)) >
abs(theta1) - limTheta)
        %disp('El segundo se corresponde con el primero');
        rho1 = rhoHough(2); theta1 = thetaHough(2);
        % Comprobar ahora si la primera recta se corresponde con la
        % segunda del frame anterior.
        if (abs(rhoHough(1)) < abs(rho2) + limRho && abs(rhoHough(1)) > abs(rho2) -
limRho) &&...
                (abs(thetaHough(1)) < abs(theta2) + limTheta && abs(thetaHough(1)) >
abs(theta2) - limTheta)
            %disp('El segundo se corresponde con el primero');
            rho2 = rhoHough(1); theta2 = thetaHough(1);
        else
            % En caso de que la segunda linea no se corresponda, se
            % debe buscar en el resto de rectas encontradas con la
            % transformada de Hough.
            fallo = 1;
            for j = 3:nPicos
                if (abs(rhoHough(j)) < abs(rho2) + limRho && abs(rhoHough(j)) >
abs(rho2) - limRho) &&...
                        (abs(thetaHough(j)) < abs(theta2) + limTheta &&
abs(thetaHough(j)) > abs(theta2) - limTheta)
%                     fprintf("El primero se corresponde con j: %d\n", j)
                    rho2 = rhoHough(j); theta2 = thetaHough(j);
                    %                           pause;
                    break;
                end
            end
            %disp('Error');
        end
```

3. Ninguna de las dos primeras rectas actuales se corresponde con la primera recta del fotograma anterior.
    a. Se considera fallo, se pasa a buscar en el resto de rectas aquellas que coincidan con las rectas del fotograma anterior.

```
    else
        fallo = 1;
        % Si no se encuentra el valor requerido en las dos primeras,
        % buscar en el resto de rectas encontradas. Se itera en el resto de
        % picos, para obtener aquel que tenga el valor más proximo al
        % anterior.
        for j = 3:nPicos
            if (abs(rhoHough(j)) < abs(rho1) + limRho) && abs(rhoHough(j)) >
abs(rho1) - limRho &&...
                    (abs(thetaHough(j)) < abs(theta1) + limTheta &&
abs(thetaHough(j)) > abs(theta1) - limTheta)
                if (abs(rhoHough(j)) - abs(rhoCorrecta(1)) <  abs(rho1) -
abs(rhoCorrecta(1))) &&...
                        (abs(thetaHough(j)) - abs(thetaCorrecta(1)) <  abs(theta1) -
abs(thetaCorrecta(1)))
                    rho1 = rhoHough(j); theta1 = thetaHough(j);
```

```
                end
            end
            if (abs(rhoHough(j)) < abs(rho2) + limRho) && abs(rhoHough(j)) >
abs(rho2) - limRho &&...
                    (abs(thetaHough(j)) < abs(theta2) + limTheta &&
abs(thetaHough(j)) > abs(theta2) - limTheta)
                if (abs(rhoHough(j)) - abs(rhoCorrecta(2)) <  abs(rho2) -
abs(rhoCorrecta(2))) &&...
                        (abs(thetaHough(j)) - abs(thetaCorrecta(2)) <  abs(theta2) -
abs(thetaCorrecta(2)))
                    rho2 = rhoHough(j); theta2 = thetaHough(j);
                end
            end
        end
    end

    % Actualizar valores de rho y theta
    rhoCorrecta = [rho1 rho2];
    thetaCorrecta = [theta1 theta2];
end
end
```

pr3_Obtener_plantillas.m

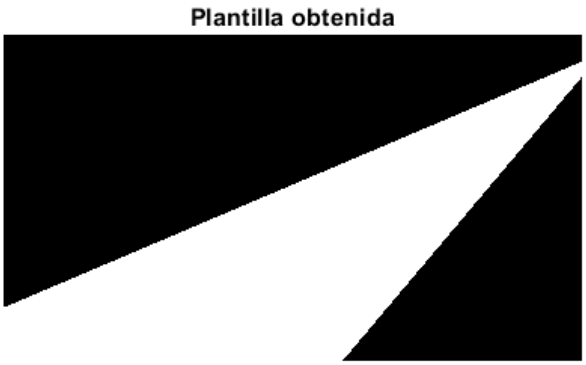
Función que recibe los valores corregidos de rho y theta y calcula una plantilla en la que se encuentra la calzada aislada.

El procedimiento que se sigue es el de comprobar si cada pixel de la imagen se encuentra comprendido entre las dos rectas, utilizando operaciones en coordenadas polares. Se deja fijo el valor de theta, y se calcula el de rho utilizando la siguiente fórmula:

$$rho = x \cdot \cos(theta) + y \cdot \sin(theta)$$

Donde **x**, **y** se corresponde con las coordenadas de cada píxel de la imagen.

```
function [fPlantilla] = pr3_Obtener_Plantillas(nFilas,nColumnas, thetaCorrecta,
rhoCorrecta)
%f1_Obtener_Plantillas Función que recibe el numero de filas y columnas y
%devuelve la mascara con la carretera aislada.

% Crear las plantillas con el tamano deseado.
fPlantilla = zeros(nFilas, nColumnas);

% Se recorre cada pixel de la imagen, y se calcula el valor de rho para
% cada una de las rectas, dejando fijas sus thetas correspondientes.
for x = 1:nColumnas
    for y = 1:nFilas
        theta1 = thetaCorrecta(1);
        theta2 = thetaCorrecta(2);
        rho1 = x * cos(theta1) + y * sin(theta1);
        rho2 = x * cos(theta2) + y * sin(theta2);

        % Una vez obtenido theta, basta con comprobar si rho se encuentra
        % entre las dos rectas.
        if ((rho1 > rhoCorrecta(1) && theta1 > 0) || (rho1 < rhoCorrecta(1) && theta1
< 0))...
                && ((rho2 < rhoCorrecta(2) && theta2 > 0) || (rho2 > rhoCorrecta(2)
&& theta2 < 0))
            fPlantilla(y,x) = 1;
        end
    end
end
end
```



pr3_Calculo_Intersecciones.m

Función que recibe un par de valores de rho y theta y calcula las intersecciones con los bordes de la imagen en coordenadas polares.

Se comienza definiendo los valores en coordenadas polares de cada una de las rectas de los bordes de la imagen. A continuación se calculan las intersecciones según la siguiente fórmula:

$$x = \frac{rho_{Linea} \cdot \sin(theta_{Borde}) - rho_{Borde} \cdot \sin(theta_{Linea})}{cos(theta_{Linea}) \cdot \sin(theta_{Borde}) - \cos(theta_{Borde}) \cdot \sin(theta_{Linea})}$$

$$y = \frac{rho_{Borde} \cdot \cos(theta_{Linea}) - rho_{Linea} \cdot \cos(theta_{Borde})}{cos(theta_{Linea}) \cdot \sin(theta_{Borde}) - \cos(theta_{Borde}) \cdot \sin(theta_{Linea})}$$

Donde **Linea** se corresponde con la línea de interés y **Borde** con cada uno de los bordes.

Una vez se tienen dos intersecciones válidas (no infinitas), se para de calcular.

```
function interseccion = pr3_Calculo_Intersecciones(rhoHough,thetaHough,nFilas,
nColumnas)
%f1_Calculo_Intersecciones Funcion que calcula la intersecciones con los
%bordes de la imagen

% Vectores que contienen los valores rho y theta de cada uno de los bordes
% de la imagen
rhoLineasBorde =    [0,       nFilas,   0,  nColumnas];
thetaLineasBorde =  [pi/2,   pi/2,      0,  0];

interseccion = [];
% Por cada linea (2 en cada caso).
for k = 1:2
    cont = 0;
    % Por cada posible interseccion
    for l = 1:4
        % Calcular la interseccion, redondear para obtener el valor en
        % pixel
        interseccionX= round((rhoHough(k)*sin(thetaLineasBorde(l)) -
rhoLineasBorde(l)*sin(thetaHough(k))) / ...
            (cos(thetaHough(k)) * sin(thetaLineasBorde(l)) - cos(thetaLineasBorde(l))
* sin(thetaHough(k))));
        interseccionY = round((rhoLineasBorde(l)*cos(thetaHough(k)) -
rhoHough(k)*cos(thetaLineasBorde(l))) / ...
            (cos(thetaHough(k)) * sin(thetaLineasBorde(l)) - cos(thetaLineasBorde(l))
* sin(thetaHough(k))));
```

```
        % Comprobar que el resultado de la interseccion no resulta ser
        % infinito (lineas paralelas a alguno de los bordes de la imagen).
        if (interseccionX ~= Inf && interseccionY ~= Inf)
            interseccion = [interseccion; interseccionX interseccionY];
            cont = cont + 1;
        end

        % Una vez se obtienen dos intersecciones, parar de calcular.
        if cont == 2
            break;
        end
    end
end
end
```

pr3_DibujaLineas.m

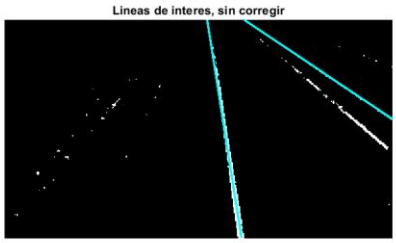
Función que recibe las intersecciones de las líneas con los bordes de la imagen y las dibuja. Hace uso de variables persistentes, para poder borrar las líneas de la imagen anterior antes de dibujar las del fotograma actual.

```
function pr3_DibujaLineas(fTratada,interseccion, recta_a_pintar, indice, p,
primerFrame)
%f1_Dibuja Funcion que dibuja las rectas sobre la imagen original

% b1 y b2 son variables en las que se guarda el plot de las lineas de una
% iteracion a otra, permite borrar exclusivamente las lineas para poder
% dibujar las del siguiente fotograma.
persistent recta1 recta2

% Vector para los colores de la primera y segunda linea de hough.
colores = ["red" "green"];

% Si no se trata del primer frame, borrar las lineas del frame anterior
if indice > primerFrame
    if (recta_a_pintar) == 0
        delete(recta1);
    else
        delete(recta2);
    end
end

% Mostrar la imagen tratada
set(p, 'CData', fTratada);
hold on

% Dibujar cada una de las lineas, seguns se trate de la imagen con lineas
% sin corregir o corregidas.
for a = 0:1
    if recta_a_pintar == 0
        recta1(a+1) = plot(interseccion(2*a + 1:2*a +2,1), interseccion(2*a + 1:2*a
+2,2),'LineWidth', 2, 'Color', 'cyan');
    else
        recta2(a+1) = plot(interseccion(2*a + 1:2*a +2,1), interseccion(2*a + 1:2*a
+2,2),'LineWidth', 2, 'Color', colores(a+1));
    end
end
hold off
end
```

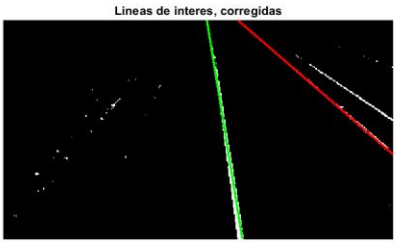
Resultados

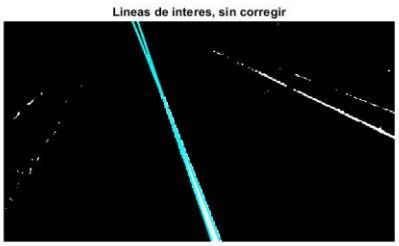
Errores

El programa falla un total de 8 veces, teniendo por tanto un porcentaje de acierto del 98.49%. A continuación se muestran aquellos fotogramas en los que falla, así como la salida del programa por la ventana de comandos.

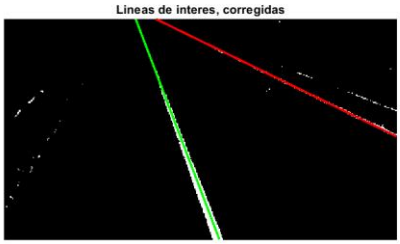


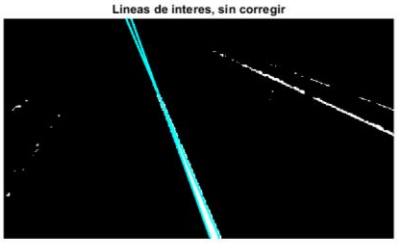
Error. Pulse cualquier tecla para continuar



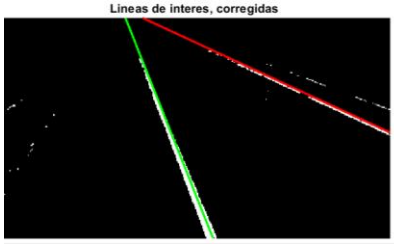


Error. Pulse cualquier tecla para continuar



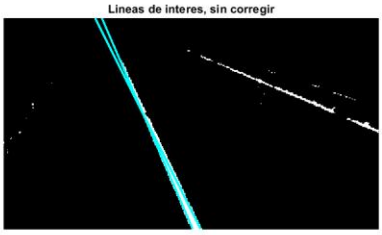


Error. Pulse cualquier tecla para continuar



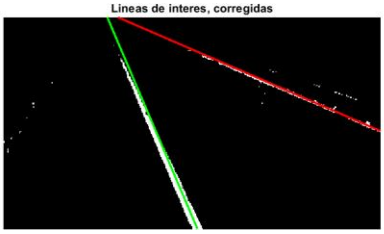
Fotograma actual: 000761.jpg

Lineas de interes, sin corregir



Error. Pulse cualquier tecla para continuar

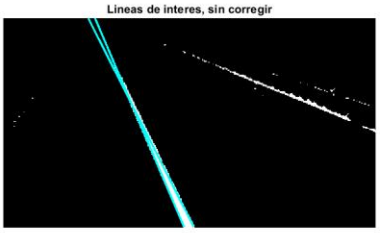
Lineas de interes, corregidas



Segmentacion de la calzada

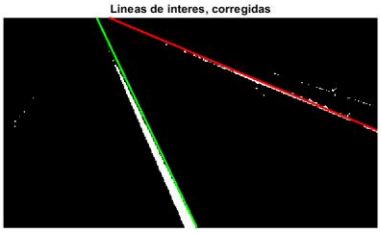
Fotograma actual: 000763.jpg

Lineas de interes, sin corregir



Error. Pulse cualquier tecla para continuar

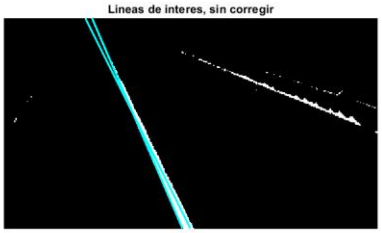
Lineas de interes, corregidas



Segmentacion de la calzada

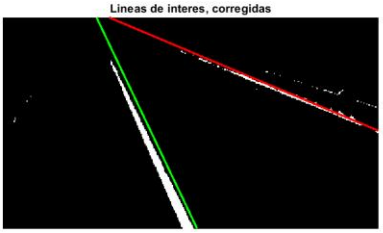
Fotograma actual: 000764.jpg

Lineas de interes, sin corregir

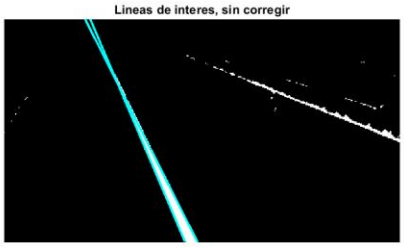


Error. Pulse cualquier tecla para continuar

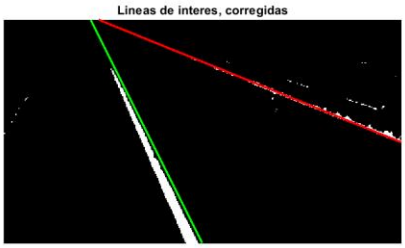
Lineas de interes, corregidas

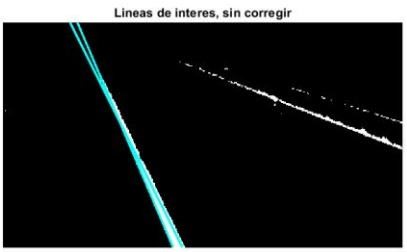


Segmentacion de la calzada

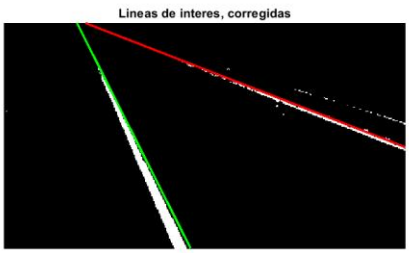


Error. Pulse cualquier tecla para continuar





Error. Pulse cualquier tecla para continuar



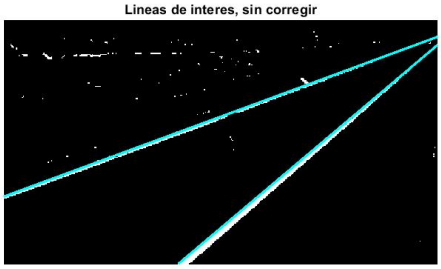
Command Window

```
Practica 3 - Sistemas de Percepcion, Pablo Leon Barriga
El programa se parara cada vez que se detecte un error, para continuar basta con pulsar cualquier tecla.
Si se quiere ralentizar la muestra de imagenes, aumentar el valor de "pausa" (valor en segundos), en la linea 6.
Se produjo un fallo en la imagen secuenciaBicicleta/000543.jpg. Contador de fallos: 1. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000749.jpg. Contador de fallos: 2. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000753.jpg. Contador de fallos: 3. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000761.jpg. Contador de fallos: 4. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000763.jpg. Contador de fallos: 5. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000764.jpg. Contador de fallos: 6. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000766.jpg. Contador de fallos: 7. Pulse cualquier tecla para continuar.
Se produjo un fallo en la imagen secuenciaBicicleta/000769.jpg. Contador de fallos: 8. Pulse cualquier tecla para continuar.
Se produjeron 8 fallos de un total de 530 imagenes. Porcentaje de aciertos: 98.49%. Porcentaje de fallos: 1.51%
fx >>
```

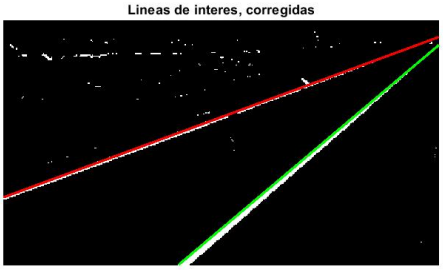
Aciertos

Fotograma actual: 000277.jpg

Lineas de interes, sin corregir

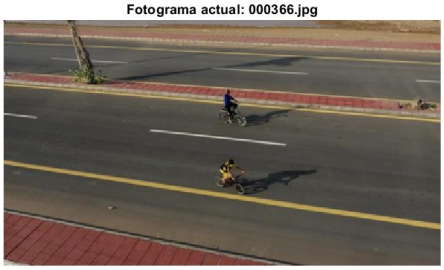


Lineas de interes, corregidas

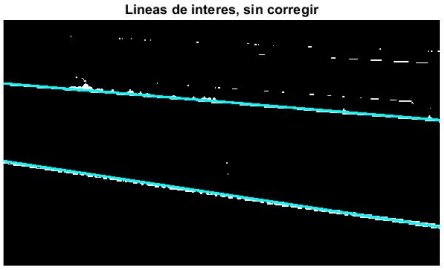


Segmentacion de la calzada

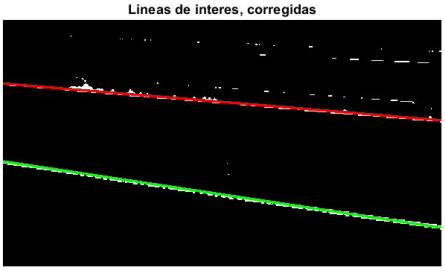
Fotograma actual: 000366.jpg



Lineas de interes, sin corregir



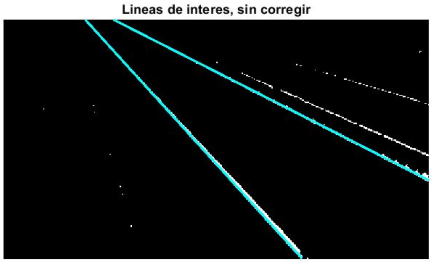
Lineas de interes, corregidas



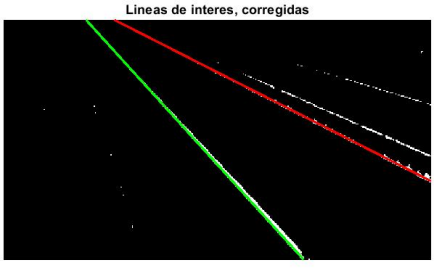
Segmentacion de la calzada

Fotograma actual: 000485.jpg

Lineas de interes, sin corregir



Lineas de interes, corregidas



Segmentacion de la calzada