UNIVERSITAT POLITÈNICA DE VALÈNCIA

ESCOLA POLITÈCNICA SUPERIOR DE GANDIA

Grado en Tecnologías Interactivas

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS DE GANDIA

Semáforo inteligente (Detector de personas)

Jose Julio Peñaranda Jara

Jonathan Javier Hernández Ramírez

1. Introducción

Este proyecto se basa en el funcionamiento de un semáforo inteligente que analiza un paso de cebra y tras esto cuenta los coches y personas de éste, además de calcular la distancia entre el paso de cebra y los coches y los peatones. Con esta información puede tomar la decisión de encender la luz roja o verde del semáforo de peatones o de coches.

2. Descripción del código

El código de este proyecto está dividido en tres funciones que se definen en los archivos siguientes:

2.1 Deteccion.m

Script encargado de la detección de coches y peatones y la cuenta de éstos.

Lo primero es inicializar la imagen base y pasarla a escala de grises, tras esto se inicializan los detectores de peatones y vehículos.

```
Img = imgOrigen;
    Img = rgb2gray(Img); % Pasar a escala de grises

    vehicleDetector = vehicleDetectorACF('front-rear-view'); %
Detector de vehiculos
    detectorPeople = peopleDetectorACF('inria-100x41'); %
Detector de personas
```

Tras esto realizamos la detección de coches y peatones, el proceso es similar en ambas, por ello definiremos solo la parte de detección de coches. La función detect nos devuelve una array de coches y un array de puntuaciones que definen lo que se parece cada objeto detectado a un coche. Detectamos los coches de la imagen y tras esto, si hay coches en la imagen, detectamos las posiciones del array de coches cuyos objetos son similares a coches con una puntuación mayor a 4 para filtrarlos en otro array “cochesBuenos”. Por último añadimos los coches buenos detectados a la imagen y contamos el número de coches en la variable numCoches.

```
numCoches = 0;
    numPersonas = 0;

    % Deteccion vehiculos
    [cboxes,cscores] = detect(vehicleDetector,Img);
    if not (isempty(cboxes))
        indice = find(cscores > 4);
        puntuacioncochesBuenos = cscores(indice);
        cochesBuenos = cboxes(indice,:);
        Img =
insertObjectAnnotation(Img,'rectangle',cochesBuenos,puntuacionco
chesBuenos);
        numCoches = size(cochesBuenos);
    end


    % Deteccion personas
    [pboxes,pscores] = detect(detectorPeople,Img);
    if not (isempty(pboxes))
```

```
        indice2 = find(pscores > 4);
        puntuacionpeatonesBuenos = pscores(indice2);
        peatonesBuenos = pboxes(indice2,:);
        Img =
insertObjectAnnotation(Img,'rectangle',peatonesBuenos,puntuacion
peatonesBuenos);
        numPersonas = size(peatonesBuenos);

    end
```

La función finalmente devuelve el número de coches y el número de peatones

```
% Return
output = [numCoches(1); numPersonas(1)];
```

2.2 Distancia.m

Script encargado de calcular la distancia respecto a los coches y peatones para tomar la decisión de que grupo está mas cerca, los coches o los peatones. El principio del código repite el código de Deteccion.m para añadir los objetos a la imagen.

```
img= imagenBase;

    vehicleDetector = vehicleDetectorACF; % Cargamos el detector
de vehiculos
    peopleDetector = peopleDetectorACF; % Cargamos el detector
de personas

    [cboxes,cscores] = detect(vehicleDetector,img);
    if not (isempty(cboxes))
        indice = find(cscores > 3);
        puntuacioncochesBuenos = cscores(indice);
        cochesBuenos = cboxes(indice,:);
        img =
insertObjectAnnotation(img,'rectangle',cochesBuenos,puntuacionco
chesBuenos);
    end


    % Deteccion personas
    [pboxes,pscores] = detect(peopleDetector,img);
    if not (isempty(pboxes))
        indice2 = find(pscores > 3);
        puntuacionpeatonesBuenos = pscores(indice2);
        peatonesBuenos = pboxes(indice2,:);
        img =
insertObjectAnnotation(img,'rectangle',peatonesBuenos,puntuacion
peatonesBuenos);
    end
```

Con la transformada de Hough detectamos las líneas del paso de cebra

```
[H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);
    P = houghpeaks(H, 5, 'threshold', ceil(0.3*max(H(:))));
```

```
    lineasPasodeCebra = houghlines(BW, T, R, P, 'FillGap', 5,
'MinLength', 7);
```

A continuación, se muestra el proceso del cálculo de distancias. El primer bucle recorre cada una de las líneas del paso de cebra, dentro de éste hay un bucle para las personas y otro para los coches, tienen un funcionamiento similar por lo que solo explicaremos bucle dedicado a las personas. Se calcula la distancia de la línea del paso de cebra en cuestión a cada uno de los peatones con puntuación adecuada y la menor de las distancias se guarda en distPeatones. Como este proceso se realiza con cada una de las líneas, provoca que la distancia se calcule con respecto a todo el paso de cebra, guardando la menor de las distancias de los coches y la menor de las distancias de los peatones.

```
    distPeatones = 99999;
    distCoches = 99999;
    for i = 1: length(lineasPasodeCebra)

        for j = 1: length(cochesBuenos)
            distcoche = sqrt( (cochesBuenos(2,j) -
cochesBuenos(1,j)).^2 + (lineasPasodeCebra(i).point2 -
lineasPasodeCebra(i).point1).^2);
            if (distcoche < distCoches)
                distCoches = distcoche;
            end
        end

        for k = 1: length(peatonesBuenos)
            distpersona = sqrt( (peatonesBuenos(2,k) -
peatonesBuenos(1,k)).^2 + (lineasPasodeCebra(i).point2 -
lineasPasodeCebra(i).point1).^2);
            if (distpersona < distPeatones)
                distPeatones = distpersona;
            end
        end
    end
```

Si la menor de las distancias de los peatones es mayor o igual a la menor de las distancias de los coches la salida será true, lo que significa que el semáforo en cuestión estará en verde, en caso contrario la salida será false.

```
% Outputs
    if (distCoches <= distPeatones)%Verde si la distancia de los
peatones es menor
        output = true;
    else
        output = false;
    end
```

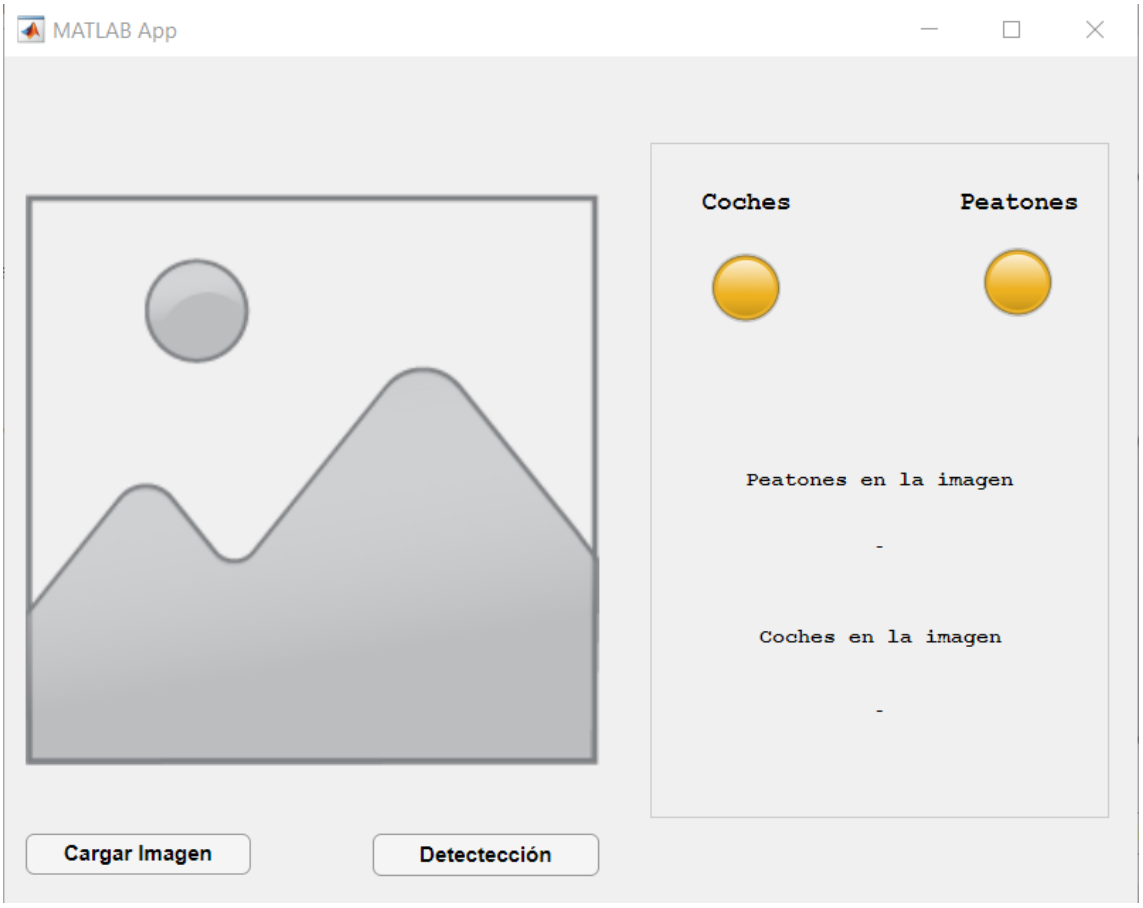
2.3 colorSemaforo.m

Script que se encarga de establecer el color del semáforo de coches (el color del semáforo de peatones será el contrario). La función recibe la condición de distancia (true si los coches están más cerca que los peatones y false si no), además del número de coches y de peatones de la imagen. La siguiente cadena de if define lo siguiente: Si hay mas coches que peatones el color será verde, si hay mas peatones que coches el color será rojo y si el número es el mismo se define por la condición de distancia.

```
function output = colorSemaforo(nCoches,nPersonas,
condicionDistancia)
    if (nCoches < nPersonas)
        output = false;%Rojo
    elseif (nCoches > nPersonas)
        output = true;%rojo
    else %Si hay el mismo numero
        output = condicionDistancia;%Verde si la distancia de
los peatones es menor
    end
end
```

3. Descripción de la interfaz

El Usuario puede cargar una imagen y al pulsar el botón Detección podrá realizar el programa. Se llama a las 3 funciones del proyecto, el número de personas y coches se muestran en los textos “nPersonasText” y “nCochesText”. Los semáforos están representados con dos Lamps al lado de la imagen. Se llama a la función Distancia y tras ello a colorSemaforo, ésta función devolverá un booleano, si es true el semáforo de coches estará en rojo y el de peatones en verde, si es false sucederá todo lo contario.



4. Referencias

- **VehicleDetectorACF**

https://es.mathworks.com/help/driving/ref/vehicledetectoracf.html

- **PeopleDetectorACF**

Detect people using aggregate channel features - MATLAB peopleDetectorACF - MathWorks España

- **App Designer**

Desarrollar apps mediante App Designer - MATLAB & Simulink - MathWorks España