Universitat Politècnica de Catalunya

Visió per Computadors

Informe Short Project: Detector de mirades

Carlota Catot Bragós

Alejandro Domínguez Besserer

Quadrimestre Tardor 2018-2019

Índex

1 Plantejament del problema

Per aquest projecte se'ns ha plantejat el següent repte: detectar en una fotografia, si una persona mira a la càmera o no. Per tal de poder resoldre aquest repte, utilitzarem mètodes de Visió de Computador amb MatLab.

2 Introducció

Al llarg els últims anys la visió per computador ha anat guanyant terreny en el món de la informàtica, és per això que moltes propostes de tractament d'imatges es defineixen amb algorismes d'aquest.

En aquest informe explicarem els diferents estats del nostre projecte, destacarem les principals complicacions i presentarem els diferents resultats obtinguts durant les diferents etapes d'aquest. Principalment hem centrat el nostre estudi a l'algorisme *HOG (Histogramof Oriented Gradients* i les propietats que aquest ens presenta, per això també hem desenvolupat una versió pròpia de l'algorisme i ha estat el descriptor principal pels nostres estudis, encara que per tal de poder realitzar-lo també hem utilitzat el descriptor *LBP (Local Binary Pattern)* (implementat per MatLab). Un cop extretes les propietats necessàries, per a entrenar al nostre sistema hem utilitzat l'algorisme de SVM (Suport Vector Machine) integrat de matlab i així crear el classificador tant d'ulls com de mirades.

A continuació presentarem un resum del funcionament del projecte, posteriorment, durant l'apartat del procediment, explicarem les diferents fases i estats que hem tingut durant el desenvolupament i també l'estudi dels resultats extrets al llarg d'aquest.

3 Funcionament general del projecte

El funcionament general del projecte ve descrit al codi *main.mlx* on partint d'una imatge d'entrada, el programa indicarà si hi ha ulls a la imatge i si aquest miren a la càmera. Per tal de fer funcionar el programa únicament és necessari tenir instal·lat a l'ordinador i executar el fitxer *main.mlx*. Si es vol canviar la imatge d'entrada, únicament cal canviar el paràmetre *imatge* del codi per la imatge desitjada.

4 Procediment seguit

Per tal d'explicar el procediment seguit al llarg del projecte, dividirem aquesta secció en 3 apartats, ja que el desenvolupament del projecte ha estat marcat per 3 fases. La primera ha estat la **preparació dels DataSets**, a la segona hem estat preparant els **classificadors**, tant d'ulls com de mirades i a la tercera hem creat el codi que per la **detecció en imatges** reals. Als annexes hem posat la versió final de cadascuna de les funcions explicades a continuació.

4.1 Preparació dels DataSets

Per començar el projecte hem preparat un seguit de DataSets per tal de començar a fer els classificadors, en total crearem 3 datasets: **TrainData**, **TestData** que seran les imatges utilitzades pel training i pel testing respectivament, i per últim **GazeLabelsData** que serà utilitzat pel classificador de mirades.

Com a dataset original, partim d'un conjunt d'imatges de prova extretes de la web de BioID (https://www.bioid.com/About/BioID-Face-Database). En cadascuna de les imatges d'aquest conjunt apareix una persona fotografiada des d'una càmera de videoconferència, amb diferents posicions de cap i mirant a diferents punts del pla o amb els ulls tancats (alguns mirant a la càmera, que són els que hem de trobar). Per cada imatge del conjunt també disposem d'un fitxer .eye que ens indicarà on es troba el centre dels ulls per a cada imatge i ens ajudarà a la creació dels datasets de Train i Test.

Primera iteració

Tant pel **TrainData** com pel **TestData** hem creat dues matrius: La primera, amb imatges d'ulls (figura 1), extretes del dataset original, on per tal d'extreure la part dels ulls, hem utilitzat les coordenades del fitxer .eye, i hem retallat amb una mida de 32x128 aquella zona de la imatge. La segona, amb imatges de no ulls (figura 2), extretes també del dataset original, però de la part de la imatge on no hi ha ulls, per tal de realitzar-ho hem extret imatges de la mateixa mida que a la matriu anterior (32x128) però amb la part de les imatges que no contenen ulls.

Per poder disposar d'un 5% d'imatges d'ulls i d'un 95% d'imatges de no ulls a la suma dels dos datasets, hem multiplicat el nombre d'imatges per 19. D'aquesta manera en tenir moltes més imatges de no ulls que d'ulls el nostre classificador podrà ser més precís. Un cop fets els càlculs del nombre d'imatges que necessitem, hem dividit les matrius en els dos datasets, utilitzat el 90% de les imatges pel TrainData (1.368 imatges d'ulls i 26.008 imatges de no ulls) i el 10% pel TestData (153 imatges d'ulls i 2891 imatges de no ulls).

Pel dataset de **GazeLabelsData** hem utilitzat l'excel *Miram.xlsx* on està indicat quines de les imatges del dataset original miren a la càmera i quines no. Això serà usat a l'apartat de classificació de mirades.

Figura 1: Exemple imatge Ulls

Figura 2: Exemple imatge NoUlls

Segona iteració

La *segona iteració* d'aquesta funció ve donada quan, en arribar a la part de detecció d'imatges completes, la finestra lliscant usada per trobar els ulls no és prou precisa, ja que es dóna al cas que només agafi un ull (es pot veure a la imatge 14), i per tant, com els training han estat realitzats amb dos ulls, resulta poc precís.

Per tal de solucionar aquest problema, hem decidit replantejar els datasets de **TrainData** i **TestData** de manera que les imatges dels ulls només tinguin un ull, no dos. Per tant, ara la mida de les imatges (tant per imatges d'ulls (figura 3) com per imatges de no ulls (figura 4)) són de mida 32x48 i la quantitat d'imatges a utilitzar per als dos datasets es veu multiplicada per dos, ja que ara en lloc de 1521 imatges d'ulls hi haurà 3042.

Com a la versió anterior, hem utilitzat un 5% de les imatges per ulls (3042 imatges) i un 95% per no ulls (57798 imatges), i també igual que a la versió anterior, hem utilitzat el 90% de les imatges pel TrainData (2.737 imatges d'ulls i 52.017 imatges de no ulls) i el 10% pel TestData (305 imatges d'ulls i 5.781 imatges de no ulls).

Pel dataSet de **GazeLabelsData** hem hagut de fer un canvi per tal de duplicar les entrades, ja que han d'haver-hi tantes entrades com imatges d'ulls. Per tal de fer això hem duplicat tots els valors donats a l'excel, d'aquesta manera el vector queda amb la mateixa mida que les imatges totals del dataset original per dos.

Figura 3: Exemple imatge Ull

Figura 4: Exemple imatge NoUll

4.2 Classificadors

Per crear i entrenar els classificadors hem utilitzat diferents algorismes d'extracció de propietats (descriptors). Principalment hem usat l'algorisme **HOG (Histogram of Oriented Gradients)** per extreure les característiques de les imatges, el sistema SVM (Support Vector Machine) i el dataset TrainData (explicades al punt anterior) per crear els classificadors. Com a altres descriptors també hem usat **LBP (Local Binary Pattern)** i **histogrames de color**, no tots ells han donat bons resultats però han estat presents durant l'estudi dels classificadors.

Un cop realitzats el classificador mitjançant l'entrenament, hem provat la seva eficiència amb les imatges de TestData i amb el crossvalidation utilitzant la funció de matlab *crossval* a totes les iteracions, per tal de comprovar que les característiques utilitzades han estat prou bones.

A continuació s'explica en detall l'estudi de les diferents versions de codi que s'ha realitzat pel classificador d'ulls i pel de mirades, ja que no hem utilitzat els mateixos descriptors per un que per l'altre. Per les versions definitiva de cadascun dels classificadors s'ha fet un estudi més extens.

4.2.1 Classificador d'ulls

Primera iteració

Durant la *primera iteració* del classificador, hem utilitzat la versió de HOG integrada a matlab per tal d'extreure les característiques de cada imatge. Al realitzar el test, el predictor treu la matriu de confusió de la imatge 5 on podem veure que hi ha un **100% d'encert**, i amb el cross validation dona un **0.0256% de taxa de generalització** Durant aquesta iteració s'han usat els datasets de les imatges amb 2 ulls (32x128)

Segona iteració

Durant la *segona iteració* del classificador, hem utilitzat els mateixos datasets que a la primera iteració però enlloc d'utilitzar el HOG integrat per matlab hem utilitzat un HOG amb codi propi, i al realitzar el testing el predictor treu la matriu de confusió de la imatge 6 on podem veure que hi ha un 100% d'encert a les imatges d'ulls però un 99.93% d'encert a les de no ulls, el que fa un **99.9965% d'encert** general, i amb el cross validation dona un **0.0417% de taxa de generalització**

Tercera iteració

Durant la *tercera iteració* del classificador, no hem realitzat cap canvi al classificador, però si als datasets. Ara les imatges son d'un sol ull, i al igual que a la iteració anterior hem utilitzat com a descriptor el HOG propi. Al realitzar el testing el predictor treu la matriu de confusió de la imatge 7 on podem veure que hi ha un **98.69% d'encert** a les imatges d'ulls però un **99.86% d'encert** a les de no ulls, el que fa un **99.8028% d'encert** general, i amb el cross validation dona un **0.4215% de taxa de generalització**

Conclusió i estudi estadístic del descriptor

Hem decidit no posar cap altre descriptor per calcular el classificador, ja que en realitzar proves només hem aconseguit un decrement de l'índex d'encert o que es quedés igual, és per això que deixem la tercera iteració com el codi definitiu, només utilitzant HOG tot i que també hem realitzat altres estudis estadístics, com per exemple, al provar d'utilitzar *histogrames de color* ens va sortir un encert general del 58.44%, aquesta versió l'hem descartat ràpid i hem decidit usar el descriptor *LBP* que ens dóna el mateix resultat que sense ell, per tant tampoc l'usarem.

Per tal de comprovar com de bo és el nostre HOG hem també realitzat una prova del codi sense les característiques HOG, només utilitzant LBP, això ens ha donat un **94.9885% d'encert** general i la matriu de confusió de la imatge 8, podem veure que el comportament general és que no detecta cap ull, per tant, encara que el percentatge d'encert és prou alt, podem dir que en aquest cas no és prou basar-se només amb aquest percentatge, sinó que és necessari mirar la matriu de confusió per tal de veure que el classificador no és útil, ja que sempre detecta el mateix i té un rati de generalització (calculat mitjançant el cross validation) de 4.9987%.

La nostra conclusió pel que afecta aquest classificador, després dels estudis anteriors és que veient que la taxa de generalització ha augmentat i el rati d'encert està a un 99% amb les característiques de HOG, deixem el classificador en aquest estat (el codi es pot veure al Annexe A.2.2). Com a dificultat a destacar durant el desenvolupament del classificador, ha estat que les diferents funcions podien arribar a tardar 45 minuts a executar-se, el que feia que el ritme de feina és vegues afectat.

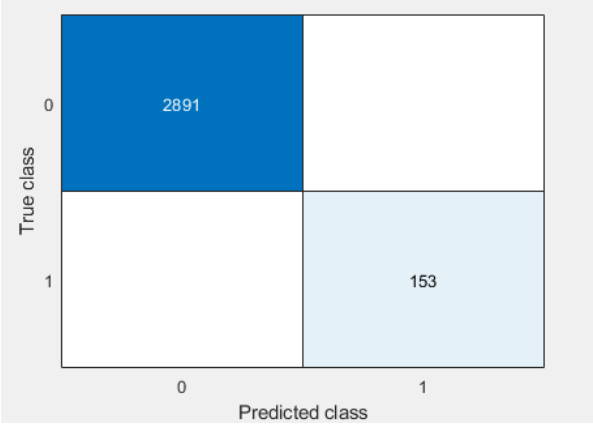

Figura 5: Matriu de confusió iteració 1

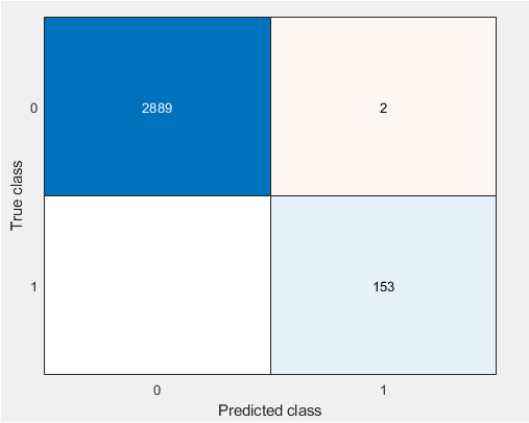

Figura 6: Matriu de confusió iteració 2

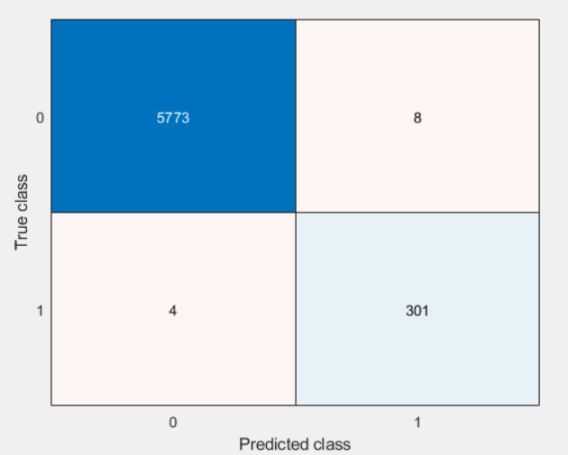

Figura 7: Matriu de confusió iteració 3

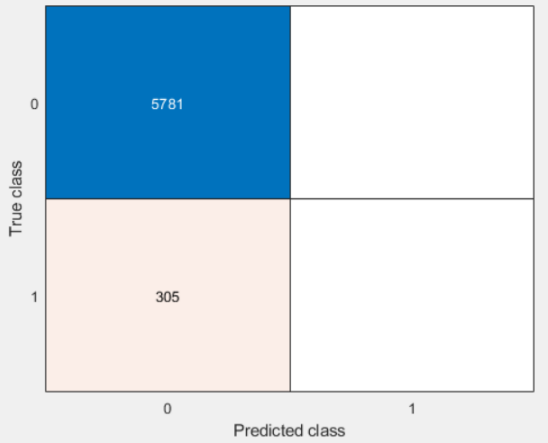

Figura 8: Matriu de confusió sense HOG

4.2.2 Classificador de mirades

Primera iteració

Durant la *primera iteració* del classificador, hem usat (al igual que a la primera iteració del classificador d'ulls) l'algorisme HOG implementat per matLab, i amb el dataSet **GazeLabelsData** hem pogut saber els labels per cada imatge. En realitzar el testing, el predictor treu la matriu de confusió de la imatge 9 amb un 93.33% d'encert per la no mirada i un 80.49% per la mirada, el que fa un **83.01% d'encert** general. Amb el crossvalidaton ens dóna que tenim un **16.45% de taxa de generalització**. Per aquesta iteració hem utilitzat les imatges de mida 32x128.

Segona iteració

Durant la *segona iteració* del classificador, utilitzat el mateix dataset que a la iteració anterior, canviant l'algorisme HOG a la versió desenvolupada per nosaltres. En realitzar el testing, el predictor treu la matriu de confusió de la imatge 10 amb un **83.33% d'encert per la no mirada** i un **76.42% per la mirada**, el que fa un **77.78%** d'encert general. Amb el crossvalidaton ens dóna que tenim un **17.32% de taxa de generalització**.

Tercera iteració

Durant la *tercera iteració* del classificador, hem utilitzat exactament el mateix codi que a la iteració anterior, utilitzant l'algorisme HOG propi, però pels nous conjunts de DataSets que només agafa un ull. Amb aquesta versió del codi, el predictor treu la matriu de confusió de la imatge 11 amb un **85.31% d'encert per la mirada** i un **73.33% per la no mirada**. Amb el crossvalidaton ens dóna que tenim un **20.90% de taxa de generalització**.

Conclusió i estudi estadístic del descriptor

Hem intentat millorar el nostre codi mitjançant altres descriptors, a l'afegir histogrames de color, el nostre rati d'encert ha estat molt inferior pel qual també hem descartat aquest descriptor per aquest classificador. Posteriorment hem provat amb l'algorisme LBP implementat per matlab i hem comprovat que dóna un resultat no molt diferent, el predictor treu la matriu de confusió de la imatge 12 amb un 86.53% d'encert per la mirada i un 70% per la no mirada, el que fa un **83.28% d'encert general**, pel que hem decidit quedar-nos amb el codi de la tercera iteració com a codi final.

Per tal de comprovar com de bo es el nostre HOG hem realitzat una prova sense les característiques HOG, només utilitzant LBP, això ens ha donat un **80.3279% d'encert general** i la matriu de confusió de la imatge 13, podem veure que el percentatge d'encert és prou alt, però observant la matriu de confusió podem veure que només és bo per les mirades, ja que en el cas de mirades dona un accent del 100% i en el de les no mirades d'un 0%. Així que igual que pel classificador d'ulls podem dir que mirant únicament el percentatge d'encert no és suficient per determinar si el descriptor és bo, cal mirar també la matriu de confusió.

La nostra conclusió per aquest classificador després dels estudis anteriors, podem dir que només amb HOG ens ha servit per tenir un classificador prou bo, tot i que hem provat d'altres descriptors com SURF (que no ens va donar bons resultats), histogrames de color i LBP, hem arribat a la conclusió de què amb un 83.28% d'encert era molt difícil trobar les característiques que el fessin millor i, ja que totes les proves han sigut negatives per nosaltres, hem decidit quedar-nos amb el que millors resultats ha tret. Per aquest classificador, els problemes de temps d'execució no han existit pel que ha estat molt més dinàmic poder fer proves. (El codi utilitzat per aquesta part es troba al annexe A.2.3)

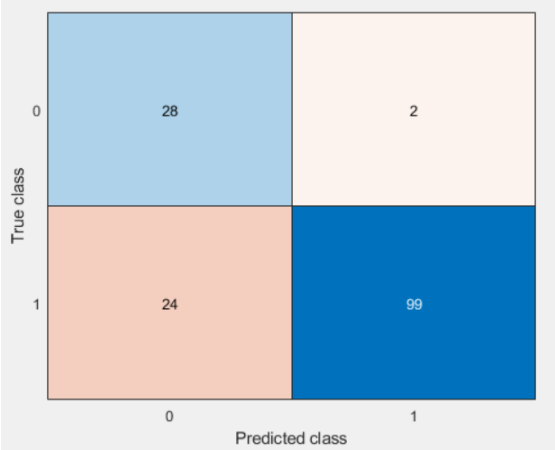

Figura 9: Matriu de confusió iteració 1

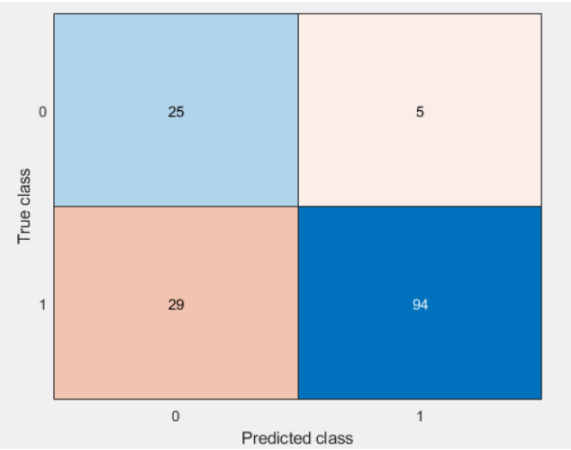

Figura 10: Matriu de confusió iteració 2

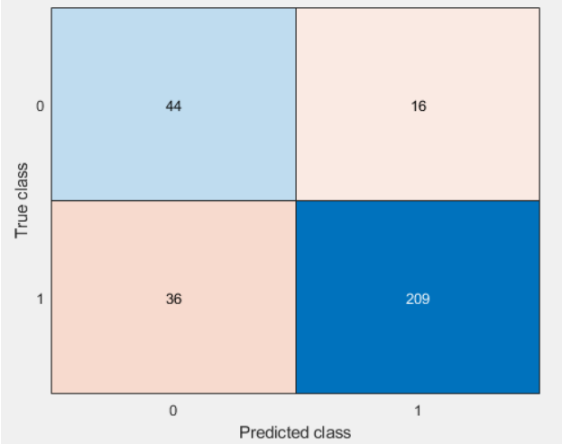

Figura 11: Matriu de confusió iteració 3

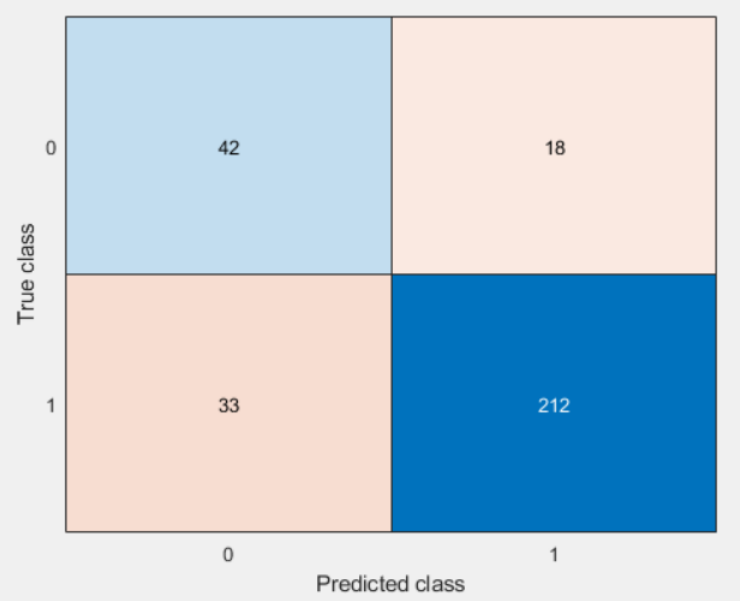

Figura 12: M.Confusió amb LBP i HOG

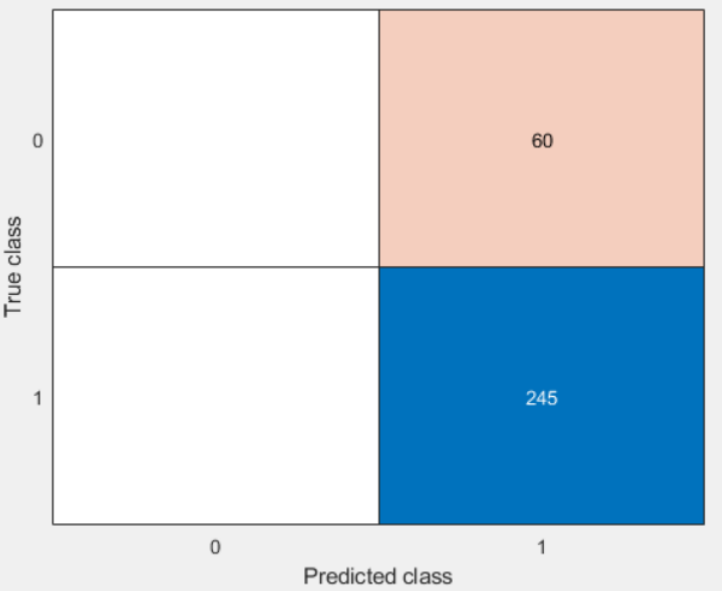

Figura 13: Matriu de confusió sense HOG

4.3 Detecció en imatges

Per últim, i després de crear els classificadors el millor que hem sapigut, hem arribat a la fase final del projecte on a partir d'imatges reals, intentarem aconseguir els ulls i un cop trobats (mitjançant el classificador d'ulls), tractarem d'identificar si la persona mira a la càmera o no (mitjançant el classificador de mirades). Marcarem els ulls amb un rectangle al voltant del mateix i la mirada amb un missatge a la pantalla.

Per realitzar aquesta funció també han calgut vàries iteracions, ja que a l'arribar a aquesta part, hem hagut de canviar codi anterior de les anteriors seccions.

Primera iteració

Durant la *primera iteració* d'aquesta funcio, hem utilitzat una finestra lliscant de 32x128 per recorrer la matriu, ja que ha estat la mida de les imatges utilitzades durant el training i testing dels diferents classificadors, per cada punt de la imatge, s'ha calculat el HOG de la finestra (utilitzant el HOG propi), com es pot veure a la imatge 14 ens vem trobar masses cops amb que la finestra lliscant es topaba amb ull i detectaba com si fosin dos ulls, per tant vem replantejar el projecte el que va donar pas a la segona iteració.

Segona iteració

Durant la *segona iteració* d'aquesta funcio, i després de realitzar un canvi als datasets inicials, hem utilitzat una finestra lliscant de 32x48 per recorrer la matriu, l'equivalent al tamany d'un ull a les imatges del dataset original. Per cada punt de la imatge, s'ha calculat el HOG de la finestra (utilitzant el HOG propi), així com també s'han utilitzat els classificadors creats anteriorment per tal de llençar el predictor. Es pot veure a la imatge 15 el que hem aconseguit bons resultats, encara que també cal destacar que per arribar a aquesta solució hem hagut de tenir en compte moltes coses i el codi ha tingut varies versions d'ell mateix. També volem destacar, que a diferencia dels arxius de testing, pel detector utilitzem el score del predictor per tal de que aquest sigui més precís. (El codi es pot veure al anexe A.3)

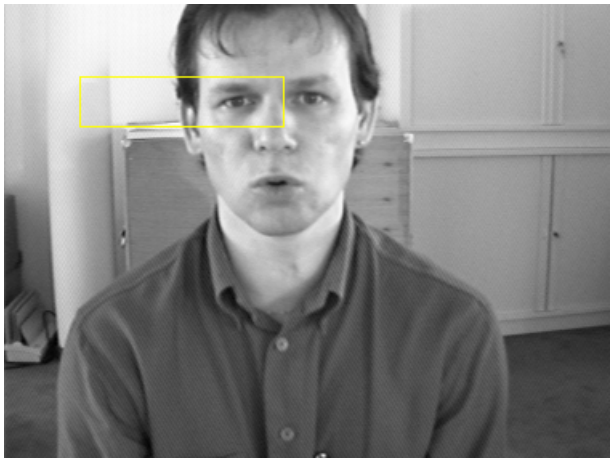

Figura 14: Finestra lliscant de 32x128

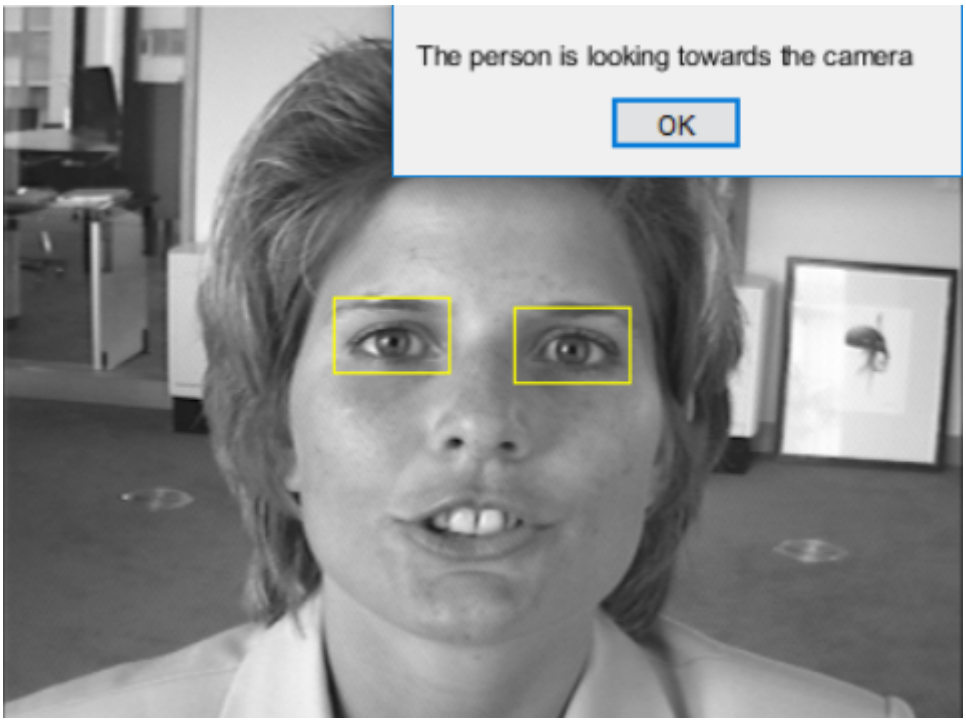

Figura 15: Resultat final

5 Conclusió

Com a conclusió podem dir que estem molt satisfets amb el que hem aconseguit. Encara i així un punt de millora seria optimitzar el SVM per tal que donés millors resultats i potser fer una cerca encara més exhaustiva de millors descriptors, tot i que hem estat capaços de desenvolupar un HOG propi i amb ell treure bons resultats.

Ens ha agradat molt poder fer aquest projecte, ja que ha sigut tot un repte poder fer un projecte d'aquesta escala amb el coneixement adquirit durant aquest quadrimestre, és per això que tot i no tenir els millors resultats, estem satisfets amb l'aprenentatge durant el procés.

A Annexes

A.1 Codi per DataSets

```
function [] = createDatasets()
    clc;
    clear;

    notEyes = zeros([32,48,1521*19*2]);
    eyeStrips = zeros([32,48,1521*2]);
    eyeCoords = zeros(1,4,1521);

    eyeLocs = dir(fullfile('data\originalDataset', '*.eye'));
    peopleImages = dir(fullfile('data\originalDataset', 'BioID*.pgm'));

    for idx = 1:numel(eyeLocs)
        fi = eyeLocs(idx);
        eyeCoordsFile = fopen(strcat(fullfile('data\originalDataset', fi.name)));
        textscan(eyeCoordsFile,'%s %s %s %s',1);
        eyeCoords(:,:,idx)= double(cell2mat(textscan(eyeCoordsFile,'%d %d %d %d',1)));
        fclose('all');
    end

    clearvars eyeLocs

    for i = 1:2:3041
        ind = ceil(i/2);
        Im = imread(fullfile('data\originalDataset', peopleImages(ind).name));
        [F C] = size(Im);
        center1 = eyeCoords(1,1:2,ind);
        center2 = eyeCoords(1,3:4,ind);
        dist = uint32(abs(center2(1)-center1(1))*0.325);
        disty = dist/2;

        left1 = max(1,center1(1)-dist);
        right1 = min(C,center1(1)+dist);
        top1 = max(1, center1(2) - disty);
        bot1 = min(F, center1(2) + disty);

        left2 = max(1,center2(1)-dist);
        right2 = min(C,center2(1)+dist);
        top2 = max(1, center2(2) - disty);
        bot2 = min(F, center2(2) + disty);


        eyeStrips(:,:,i) = imresize(Im(top1:bot1,left1:right1),[32,48]);
        eyeStrips(:,:,i+1) = imresize(Im(top2:bot2,left2:right2),[32,48]);
    end
```

```
    n = 1521;
    for i = 1:n
        Im = imread(fullfile(peopleImages(i).folder, peopleImages(i).name));
        [F C] = size(Im);
        for j = 1:38
            y = randi(F-32);
            x = randi(C-48);
            while  (x < eyeCoords(1,1,i) && eyeCoords(1,1,i) < (x+48) || ...
                    x < eyeCoords(1,3,i) && eyeCoords(1,3,i) < (x+48)) && ...
                   (y < eyeCoords(1,2,i) && eyeCoords(1,2,i) < (y+32) || ...
                    y < eyeCoords(1,4,i) && eyeCoords(1,4,i) < (y+32))
                y = randi(F-32);
                x = randi(C-128);
            end
            notEyes(:,:,(i-1)*38+j) = Im(y:y+31,x:x+47);
        end
    end

    nEyes = uint32(size(eyeStrips,3))-1;
    nNotEyes = uint32(size(notEyes,3))-1;
    trainingEyes = eyeStrips(:,:,1:nEyes*0.9);
    trainingNotEyes = notEyes(:,:,1:nNotEyes*0.9);
    testingEyes = eyeStrips(:,:,nEyes*0.9+1:end);
    testingNotEyes = notEyes(:,:,nNotEyes*0.9+1:end);

    save('data\TrainData.mat', 'trainingEyes','trainingNotEyes');
    save('data\TestData.mat', 'testingEyes','testingNotEyes');

    expectedLabels = xlsread("data\Miram.xlsx", 1, "E5:E1525");
    j = 1;
    for i = 1:1521*2
        if mod(i,2) == 0
           Labels(i,1) = expectedLabels(j);
           j = j+1;
        else
           Labels(i,1) = expectedLabels(j);
        end
    end

    save('data\GazeLabelsData.mat', 'Labels')

end
```

A.2 Codi per classificadors

A.2.1 Codi HOG propi

```
function [HOGdescriptor] = getHOG(I)
% INPUT IS AN IMAGE (MUST HAVE ROWS AND COLUMNS BE MULTIPLES OF 8,
%                       OTHERWISE IT WILL BE SCALED)
% OUTPUT IS A HOG DESCRIPTOR
    h = [-1 0 1];
    h2 = [-1;0;1];

    ndg = double(I);

    Hgrad = imfilter(ndg,h); %contorns horitzontals
    Vgrad = imfilter(ndg,h2); %contorns verticals
    %compute magnitude and angle of vectors
    MAG = hypot(Vgrad,Hgrad); % magnitude = sqrt(V^2 + H^2)
    D = Vgrad./Hgrad;
    D(isnan(D))=0;
    ANG = atand(D); %angle of each gradient vector
    ANG = mod(ANG,180);
    [F, C] = size(ndg);
    cellH = 8;
    cellW = 8;
    HOGs = cell((F/8),(C/8)); % matrix of histos of bins of pixels
    for i = 1:(F/8)
        for j = 1:(C/8)
            HOGs{i,j} = zeros(1,9,'single'); %bins of 20 deg ranges
        end
    end
    %calculate vale of each cell: histogram of gradients of pixels
    for i = 1:(F/8)
        for j = 1:(C/8)
            s = 0;
            for k = 1:cellH
                for m = 1:cellW
                    %count value of gradient into histogram of cell
                    deg =ANG(k+(i-1)*cellH,m+(j-1)*cellW);
                    mag =MAG(k+(i-1)*cellH,m+(j-1)*cellW);
                    %decide contribution to bins of deg
                    %bins are 10 30 50 70 90 ....
                    offset = mod(deg,20) - 10;
                    %contributions to adjacent bins
                    %one must always be 0
                    bin = ceil(deg/20);
                    if(bin==0)
                        bin = 1;
                    end
                    [left,right,center]=binContribs(offset);
```

```
                    if(bin == 1)
                        HOGs{i,j}(9) = HOGs{i,j}(9) + left*deg;
                        HOGs{i,j}(bin) = HOGs{i,j}(bin) + center*deg;
                        HOGs{i,j}(bin+1) = HOGs{i,j}(bin+1) + right*deg;
                    elseif(bin == 9)
                        HOGs{i,j}(bin-1) = HOGs{i,j}(bin-1) + left*deg;
                        HOGs{i,j}(bin) = HOGs{i,j}(bin) + center*deg;
                        HOGs{i,j}(1) = HOGs{i,j}(1) + right*deg;
                    else
                        HOGs{i,j}(bin-1) = HOGs{i,j}(bin-1) + left*deg;
                        HOGs{i,j}(bin) = HOGs{i,j}(bin) + center*deg;
                        HOGs{i,j}(bin+1) = HOGs{i,j}(bin+1) + right*deg;
                    end
                end
            end
        end
    end

    %BLOCK NORMALIZATION OF overlapping 2X2 CELLS
    [n, m] = size(HOGs);
    normHOGs = cell(n-1,m-1); % matrix of normalized blocks of hists
    for i = 1:n-1
        for j = 1:m-1
            normHOGs{i,j} = zeros(1, 36, 'single'); %bins of 20 deg ranges
        end
    end

    for i = 1:n-1
        for j = 1:m-1
            normHOGs{i,j} = [HOGs{i,j} HOGs{i,j+1} HOGs{i+1,j} HOGs{i+1,j+1}];
            normHOGs{i,j} = normHOGs{i,j} / norm(normHOGs{i,j}); %normalize
        end
    end

    %FINALLY: THE DESCRIPTOR OF THE IMAGE IS THE NORMALIZED HISTOGRAM MATRIX
    HOGdescriptor = single(cell2mat(normHOGs));
    HOGdescriptor = transpose(HOGdescriptor(:));
end
```

A.2.2 Codi per Classificador d'ulls

```
function [] = eyeTraining()
    Data = load('data\TrainData.mat');

    hog = getHOG(Data.trainingEyes(:,:,1));
    featureSize = length(hog);
    trainigEyesSize = length(Data.trainingEyes);
    trainigEyesNotSize = length(Data.trainingNotEyes);
    totalSize = trainigEyesSize+trainigEyesNotSize;
    Features = zeros(totalSize, featureSize, 'single');
    Labels = [ones(trainigEyesSize,1); zeros(trainigEyesNotSize,1)];

    for i = 1:length(Data.trainingEyes)+length(Data.trainingNotEyes)
        if (i <= length(Data.trainingEyes))
            imatge = Data.trainingEyes(:,:,i);
        else
            imatge = Data.trainingNotEyes(:,:,i-length(Data.trainingEyes));
        end
        Features(i,:) = single(getHOG(imatge));
    end

    eyeClassifier = fitcsvm(Features, Labels);
    save('data\eyeClassifier.mat', 'eyeClassifier');
end

function [] = eyeTesting()
    Data = load('data\TestData.mat');
    classifier = load('data\eyeClassifier.mat');

    for i = 1:length(Data.testingEyes)+length(Data.testingNotEyes)
        if (i <= length(Data.testingEyes))
            imatge = Data.testingEyes(:,:,i);
            expected(i,1) = 1;
        else
            imatge = Data.testingNotEyes(:,:,i-length(Data.testingEyes));
            expected(i,1) = 0;
        end
        Features(i,:) = single(getHOG(imatge));
    end

    prediction = predict(classifier.eyeClassifier, Features);

    cmatrix = confusionmat(expected, prediction);
    resultatNoUlls = 100*cmatrix(1,1) / (cmatrix(1,1) + cmatrix(1,2))
    resultatUlls = 100*cmatrix(2,2) / (cmatrix(2,1) + cmatrix(2,2))

    cchart = confusionchart(expected, prediction);
end
```

A.2.3 Codi per Classificador de mirades

```
function [] = gazeTraining()
    Data = load('data\TrainData.mat');
    Data2 = load('data\GazeLabelsData.mat');

    hog = getHOG(Data.trainingEyes(:,:,1));
    %featureLBP = extractLBPFeatures(Data.trainingEyes(:,:,1), 'CellSize', [8 8]);
    featureSize = length(hog);

    trainigEyesSize = length(Data.trainingEyes);
    Features = zeros(trainigEyesSize, featureSize, 'single');
    Labels = Data2.Labels(1:length(Data.trainingEyes));

    for i = 1:length(Data.trainingEyes)
        imatge = Data.trainingEyes(:,:,i);
        featureHOG = single(getHOG(imatge));
        %featureLBP = extractLBPFeatures(imatge, 'CellSize', [8 8]);
        %Features(i,:) = horzcat(featureHOG, featureLBP);
        Features(i,:) = featureHOG;
    end

    gazeClassifier = fitcsvm(Features, Labels);
    save('data\gazeClassifier.mat', 'gazeClassifier');

end

function [] = gazeTesting()
    Data = load('data\TestData.mat');
    classifier = load('data\gazeClassifier.mat');
    Data2 = load('data\GazeLabelsData.mat');
    Labels = Data2.Labels(2738:end);

    for i = 1:length(Data.testingEyes)
        imatge = Data.testingEyes(:,:,i);
        featureHOG = single(getHOG(imatge));
        %featureLBP = extractLBPFeatures(imatge, 'CellSize', [8 8]);
        %Features(i,:) = horzcat(featureHOG, featureLBP);
        Features(i,:) = featureHOG;
    end

    prediction = predict(classifier.gazeClassifier, Features);

    cmatrix = confusionmat(Labels, prediction);
    resultatNoMira = 100*cmatrix(1,1) / (cmatrix(1,1) + cmatrix(1,2))
    resultatMira = 100*cmatrix(2,2) / (cmatrix(2,1) + cmatrix(2,2))

    cchart = confusionchart(Labels, prediction);
end
```

A.3 Codi Detecció en imatges

```
function [ImOut] = eyesDetector(Im)
    %uses a sliding window to detect a pair of eyes within an image
    %SVM HAS TO BE TRAINED BEFORE INVOKING THIS FUNCTION!
    eyeClassifier = load('data\eyeClassifier.mat');
    gazeClassifier = load('data\gazeClassifier.mat');
    %for color images
    if (size(Im,3) > 1)
        I = rgb2gray(Im);
    else
        I=Im;
    end

    [F C] = size(I);
    ImOut=Im;
    eyesFound=0;
    scores = [];
    windows=[];

    for i = 1:4:(F-31)
        for j = 1:4:(C-47)
            window = single(getHOG(I(i:i+31,j:j+47)));
            [label,score,cost] = predict(eyeClassifier.eyeClassifier, window);
            if(label==1) %eye detected
                %ImOut = insertShape(ImOut,'Rectangle',[j i 48 32]);
                eyesFound=eyesFound+1;
                scores = cat(1,scores,score(2));
                windows = [windows;[j i 47 31]];
            end

        end
    end
    if(eyesFound==0)
        f = msgbox('Input image contains no eyes','eyesDetector');
        return
    end
    %point out the best candidates to be eyes in output image

    [m,ind1] = max(scores);
    ImOut = insertShape(ImOut,'Rectangle',windows(ind1,:));
    scores(ind1)=[];
    [m,ind2] = max(scores);
    while(abs(windows(ind2,1)-windows(ind1,1)) < 24)   %try not to pick same eye twice
        scores(ind2)=[];
        [m,ind2] = max(scores);
    end
    ImOut = insertShape(ImOut,'Rectangle',windows(ind2,:));
```

```
    %detect if eyes are looking towards camera

    eye1 = single(getHOG(imcrop(I,windows(ind1,:))));

    eye2 = single(getHOG(imcrop(I,windows(ind2,:))));

    prediction1 = predict(gazeClassifier.gazeClassifier, eye1);
    prediction2 = predict(gazeClassifier.gazeClassifier, eye2);

    figure;imshow(ImOut);
    if(prediction1 || prediction2)
        f = msgbox('The person is looking towards the camera','eyesDetector');
    else
        f = msgbox('The person is NOT looking towards the camera','eyesDetector');
    end
end
```