Short Project

Detecció de la mirada mitjançant visió per computador.

Visió per Computador

Grup 12

Gonzalo Recio
Nil Mullol

Q1 - 2017-2018

FIB-UPC

Índex

1. Tractament de les dades

Pel projecte final tenim un dataset amb 1521 imatges de cares de la mateixa mida. El repte tractava de crear un classificador que ens digués si la persona de la imatge esta mirant a la càmera o no. Primer hem realitzat un classificador per a detectar un ull en una imatge i, posteriorment, un altre classificador que determinés si un ull mira a càmera o no. De cada imatge hem extret dos ulls, de manera que teníem 3042 imatges d'ulls, dels quals hem dividit, aleatòriament, 2000 per a training i 1042 per a test (proporció 70%-30% aprox.).

Abans de poder provar un classificador, aquest s'ha d'entrenar, per tal de discernir les diferents classes que necessitem, segons les característiques que trobi.

Primer classificador: Ulls o no ulls?

El primer classificador, el detector d'ulls, és l'encarregat de processar els 2000 primers ulls (escollits aleatòriament) de les imatges, a partir de les coordenades d'aquests, i extreure'n les característiques per a la fase d'aprenentatge. Però no és suficient obtenint les característiques dels ulls, doncs el nostre classificador no tindria un aprenentatge complet, especialment el coneixement sobre què no és un ull. Per tant, assumint que els 2000 ulls han de compondre només un 10% de totes les imatges que ha d'aprendre el classificador, procedim a extreure les característiques de 18000 imatges més (no ulls), de les mateixes dimensions que els ulls, per a consolidar l'aprenentatge. Aquestes imatges han estat extretes de cada una de les imatges de cares originals, fent captures aleatòries però controlant de no apropar-se als ulls, ja que l'objectiu d'aquestes imatges és que representin tot allò que no és un ull.

Un cop obtingut aquest vector de característiques per a cada imatge, procedim a entrenar el nostre classificador mitjançant la funcionalitat TreeBagger de Matlab. Aquesta funcionalitat requereix tres inputs:

- La **taula de característiques**, en la que cada fila representa un element (ull, no ull) i cada columna una característica extreta per a aquest element.
- **Vector de classes associades**, en el que indiquem per a cada una dels files de la taula de característiques, si es tracta d'un ull o d'un no ull.
- Nombre **d'arbres binaris de decisió**: El TreeBagger és un classificador basat en els Random-Forests. Cada arbre compara un nombre de característiques amb les d'una

imatge i determina la probabilitat de que pertanyi a una classe. Aquest nombre d'arbres convé fixar-lo empíricament, tot i que els valors usuals són al voltant de 100.

Un cop completat aquest aprenentatge, Matlab ens retorna el nostre classificador, el qual usarem per detectar ulls en imatges noves, diferents a les que ha après el descriptor. D'aquesta manera, procedim a extreure les característiques dels 1042 ulls restants i utilitzem la funció predict del predictor, que farà la matriu de confusió del nou input (test set) sobre l'aprenentatge anterior, permetent que cada un dels 1042 ulls rebi un score, segons les seves característiques extretes, que el categoritzarà com a ull o no ull. Com que l'input és tot ulls, el resultat ideal de la matriu de confusió en aquest cas concret, seria el següent:

	Score no ull	Score ull
Nou ull 1	0	1
Nou ull 2	0	1
Nou ull 3	0	1
...	0	1
Nou ull x	0	1

És a dir, sense falsos positius ni negatius, per al nostre testeig de 1042 ulls:

Matriu de Confusió	**Observat** **ull**	**no-ull**
Predit **ull**	**1042**	**0**
no-null	**0**	**0**

Però, degut a que un ull i un no ull probablement de forma estadística tenen alguna característica comuna, l'objectiu és que l'score afavoreixi l'ull, si l'input n'és un, o no-ull en el cas contrari. És a dir, el nostre *decision boundary* és 50% per determinar una classe o l'altra.

Per tant, un cop dissenyat aquest sistema, valorarem els resultats proporcionats per diferents característiques extretes i escollir les que ens proporcionin millor discriminació i scores.

Segon classificador: El secret està en la mirada

Amb el segon classificador volem poder discernir els ulls que miren a la càmera dels que no ho fan. Per tant, procedirem a realitzar el mateix procés d'abans. Ara el sistema ha de fer una distinció molt més subtil que abans, doncs la diferència entre un ull que mira i un ull que no és menor que entre un ull i un no ull. D'aquesta manera l'aprenentage consistirà en obtenir les característiques de 2000 ulls, però ara usarem només aquests com a aprenentage (entrats en ordre aleatori), indicant en el nou vector de classes associades quin de cada un d'aquests ulls mira i quin no ho fa. Aquest vector de classes associades l'hem obtingut gràcies a la col·laboració d'altres companys de l'assignatura. Per tant, un cop extretes les característiques de cada element i obtingut el vector que marca cada element com a ull que mira o no, procedim a usar el TreeBagger per obtenir el nou classificador.
Ara, repetim el procés que hem realitzat amb el primer classificador i procedim a extreure les característiques dels 1042 ulls restants i obtenim la matriu de confusió amb els scores que cada ull ha rebut per a cada una de les dues classes. Igual que abans, la matriu ideal seria:

	Score ull que no mira	Score ull que mira
Nou ull 1	0	1
Nou ull 2	0	1
Nou ull 3	0	1
...	0	1
Nou ull x	0	1

És a dir, per a un cas de 1042 ulls de testeig, amb 542 que miren i 500 que no:

Matriu de Confusió	**Observat** mira	**Observat** no-mira
Predit mira	**542**	**0**
Predit no-mira	**0**	**500**

Però com que ara la diferència entre un ull que mira i un que no mira és encara més subtil, les distincions són encara més complicades. Per tant, seguim classificant cada ull depenent de quina classe hagi rebut major puntuació.

Fins ara hem parlat de la metodologia que hem seguit per a realitzar la tasca de diferenciar ulls de no ulls i quins d'aquests miren. Hem estat parlant que totes les distincions s'han fet gràcies a l'extracció de característiques, i ara procedirem a parlar de quines hem usat i quins han estat els resultats obtinguts.

2. Classificador ulls vs. no ulls

Com ja hem comentat abans, per a l'aprenentatge del classificador d'ulls/no ulls utilitzem 2000 (10%) imatges d'ulls de dimensions 64x64 píxels, i 18000 (90%) imatges de no ulls, de les mateixes dimensions. El raonament inicial que vam fer és, ja que els ulls tenen una forma molt particular i destaquen molt respecte la cara, utilitzar **Historgrames de Gradients Orientats** (HOG) podia ser una bona solució per començar. Provant diferents valors per als paràmetres més rellevants de HOG (dimensions de les cel·les, nombre de bins de l'histograma, mida de blocs, orientació amb signe), que serà tractada en la següent secció, vam obtenir molt bons resultats. Però, tot i això, sabíem que aquests podien millorar.

Seguint la idea de l'article de Wang i cols.[1] (2009): "*An HOG-LBP human detector with partial occlusion handling*", en el que s'explica com treballant amb característiques HOG juntament amb Local Binary Patterns (LBP) es millora en el reconeixement de persones, vam procedir a provar aquesta combinació. Tot i que el cas que ens ocupa no és ben bé l'esmentat en l'article, el raonament de "HOG performs poorly when the background is cluttered with noisy edges. Local Binary Pattern is complementary in this aspect. It can filter out noises using the concept of uniform pattern." ens va fer pensar que podia ajudar a filtrar casos en que el soroll de fons pogués alterar el resultat.

D'aquesta manera, vam procedir a fer tests amb diferents valors dels paràmetres de LBP més rellevants (dimensions de les cel·les, radi del patró circular, ús de caracterísiques invariants a la rotació, nombre de veïns usats per píxel) per a extreure el **Patró Local Binari Uniforme** (LBP) que proporcioni millors resultats.

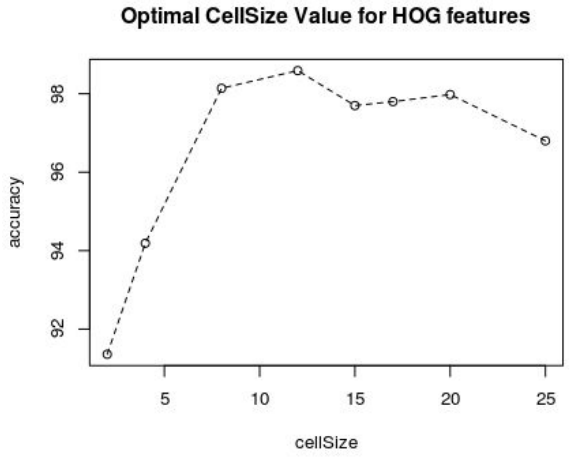
[1]Wang X, Han TX, Yan S. *An HOG-LBP human detector with partial occlusion handling.* Conference Paper in Proceedings / IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision. November 2009.

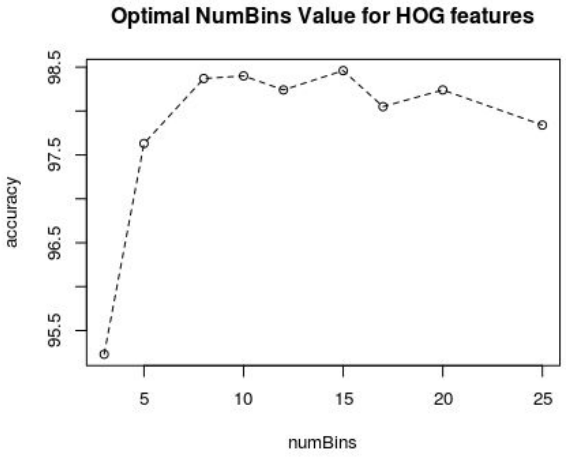
Per últim, per intentar rebre millors resultats vam utilitzar les mitjanes de les diferents matrius d'imatge per a extreure més característiques.

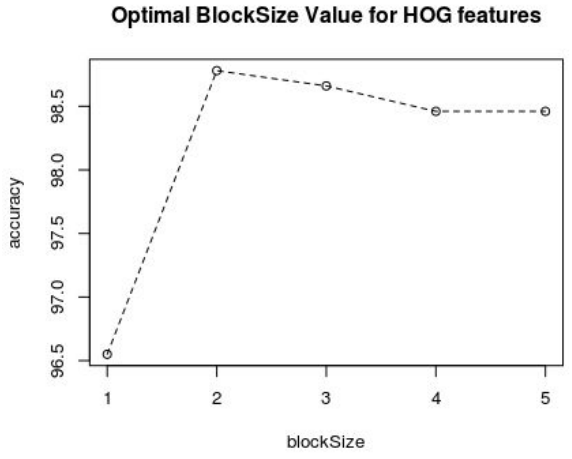
Histograms of Oriented Gradients

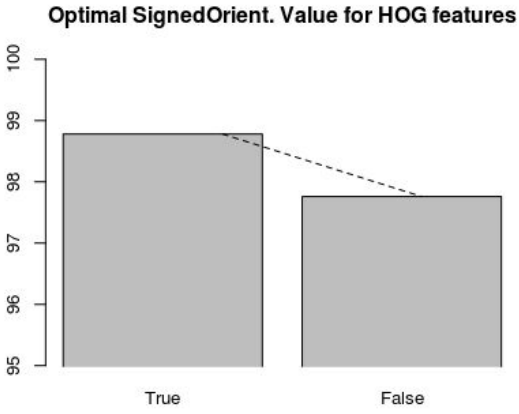
Hem intentat trobar els valors més òptims dels següents paràmetres:

- **Cell Value**: Mida del la finestra de HOG.
- **Nombre de Bins**: Obtenim orientacions més precises si l'augmentem.
- **BlockSize**: Nombre de cel·les per bloc, com més petit és aquest valor, més significància tindran els píxels locals i ajuda a suprimir els canvis d'il·luminació.
- **SignedValue**: Valors d'orientació entre 0º i 180º o entre -180º i 180º. Amb signe obtenim informació sobre el sentit de la transició de les vores de l'histograma.









Crida òptima:

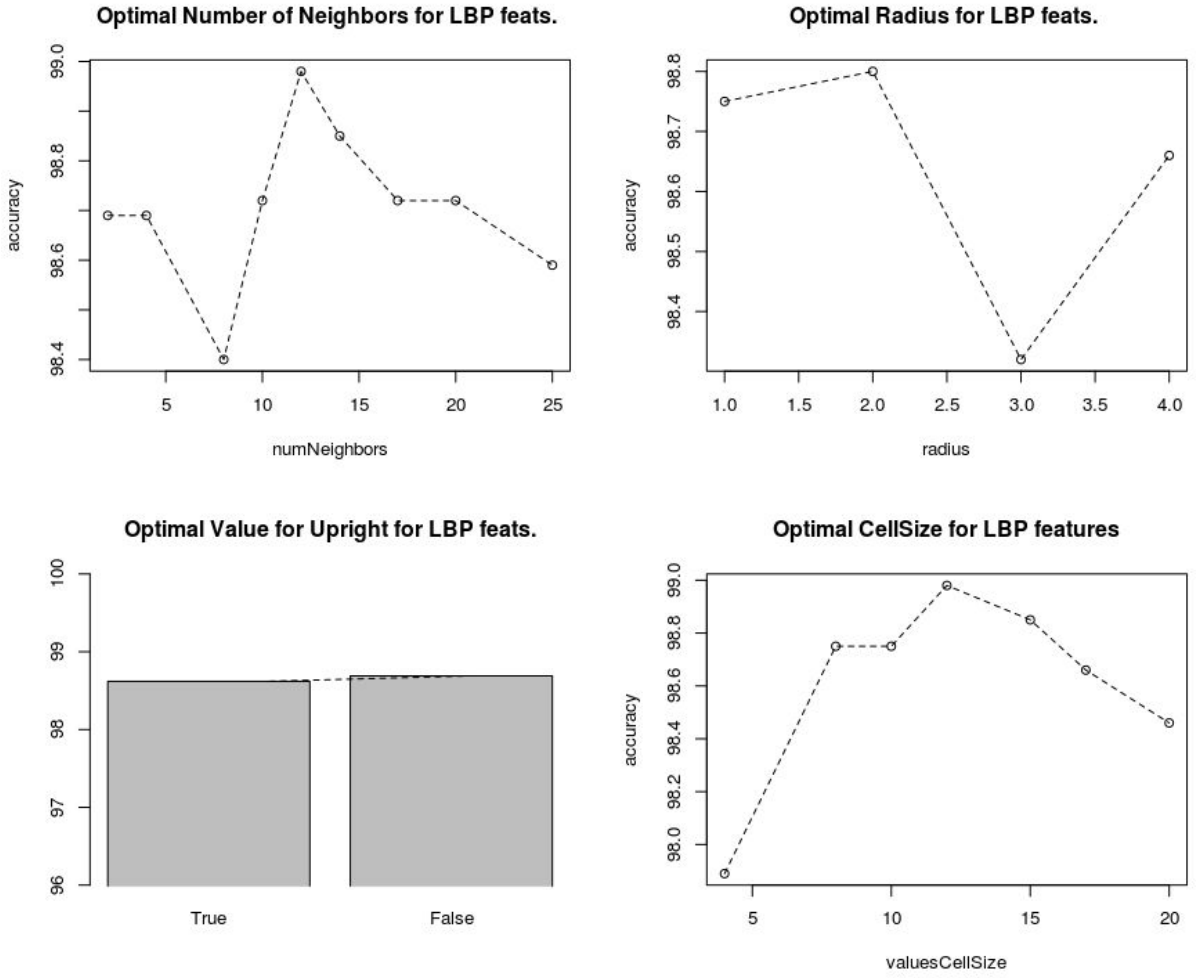
```
featuresHog = extractHOGFeatures(I,'CellSize',[12 12],
                                   'NumBins',15,
                                   'BlockSize',[2 2],
                                   'UseSignedOrientation',true);
```

Local Binary Patterns

Hem intentat trobar els valors més òptims dels següents paràmetres:

- **Cell Value:** Mida del la finestra de LBP.
- **Nombre de veïns:** El nombre de veïns que s'usaran per a computar l'LBP.
- **Radi:** Radi del patró circular per a escollir els veïns que usarà cada píxel.
- **Upright:** Permet fer que les features siguin invariants a la rotació.

Usant HOG amb els millors paràmetres, més LBP:



Crida òptima:

```
featuresLBP = extractLBPFeatures(I, 'NumNeighbors', 12,
                                    'Radius', 2,
                                    'Upright', false,
                                    'CellSize', [12 12]);
```

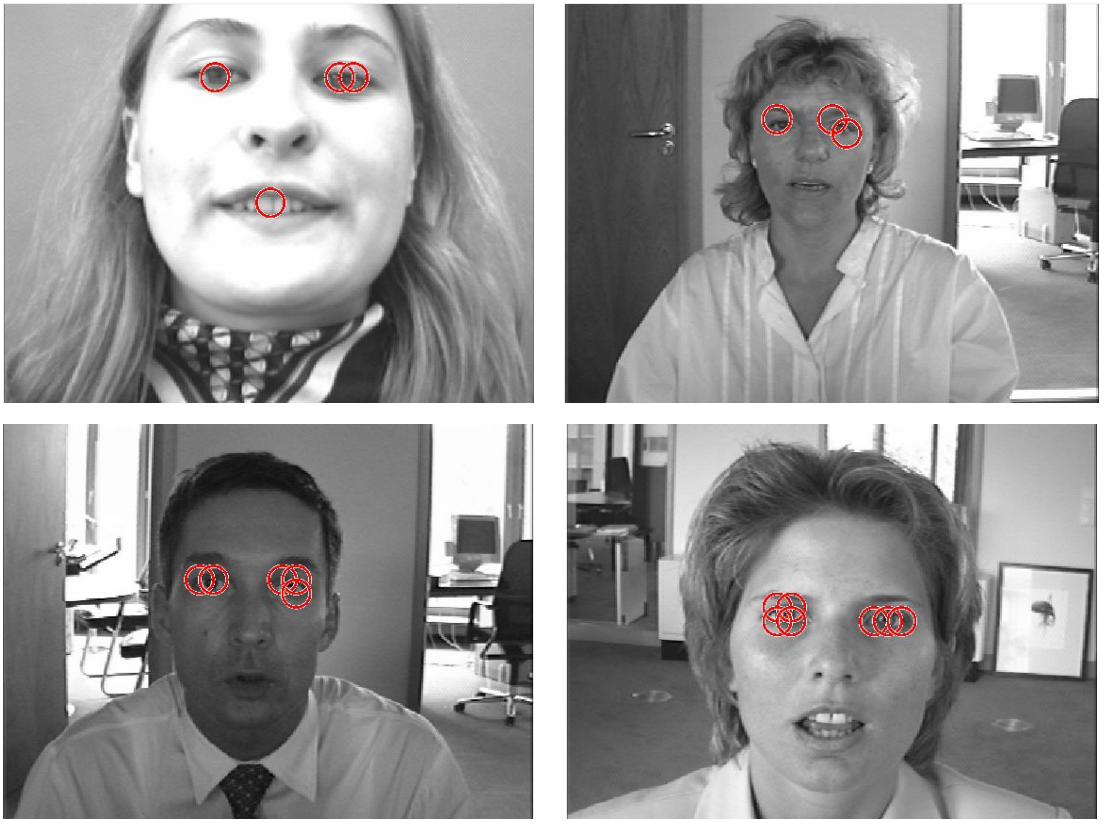
Amb aquestes features ens ha donat una precisió del **98,58**% de mitjana amb els 1042 ulls de test.

Finalment, hem decidit afegir el descriptor següent, utilitzant les mitjanes de la imatge, intentant minimitzar l'impacte de l'intensitat en el procés de decisió:

```
mcols  és la mitjana de cada columna.
mrows  és la mitjana de cada fila.
featuresMean = [mcols mrows'];
featuresMean = (featuresMean-min(featuresMean(:))) ./
               (max(featuresMean(:)-min(featuresMean(:)))); % Normalitzar
```

Aplicant aquest últim descriptor (**projecció de mitjanes**) sobre una combinació de **HOG** + **LBP** amb paràmetres optimitzats, hem obtingut un **98,91**% de precisió mitjana per als 1042 ulls de test.

Per comprovar que el nostre classificador d'ulls funciona per a detectar els ulls fins i tot sense passar-li directament les imatges d'ulls, hem fet passar una finestra de dimensions 64x64 lliscant cada 10 píxels sobre una imatge d'una cara i, com es veu a les imatges, ha trobat ulls on tocava, tot i que també tenia alguns falsos positius a vegades al passar per la boca. A les conclusions tenim més proves del lliscador amb imatges de fora del dataset.



[Funció cercles: `viscircles(center, radi, 'Color','r'); `]

3. Classificador mira vs no mira

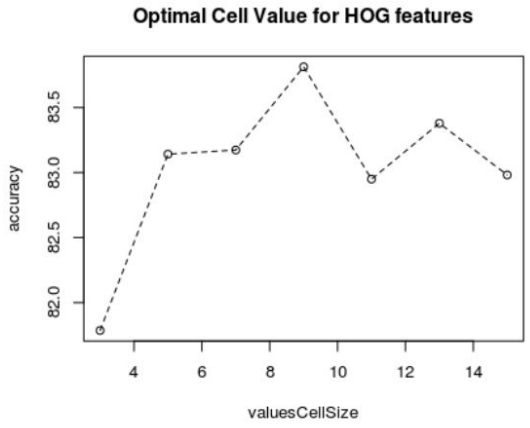
Una vegada sabem on es troben els ulls donada una imatge, procedim a detectar la mirada. Per a detectar si l'ull està mirant a càmera o no, hem utilitzat els mateixos descriptors, ja que els histogrames de gradients orientats poden donar informació de si la nineta de l'ull esta enfocant en direcció a la càmera o no: en cas de mirar directament, la nineta és més circular que en cas contrari (més circular implica que l'histograma estarà més igualat en totes direccions).

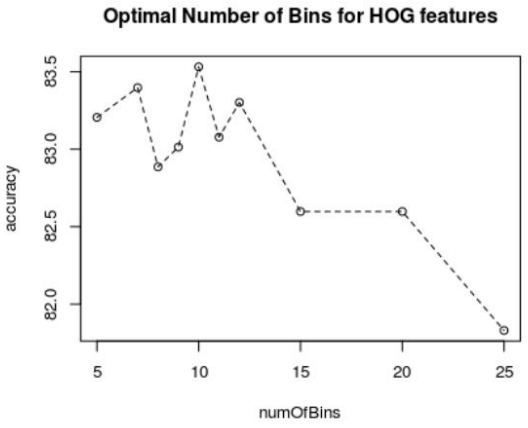
Tot i així, hem hagut de configurar els paràmetres d'extracció de característiques de manera diferent a la del classificador de ull/no ull.

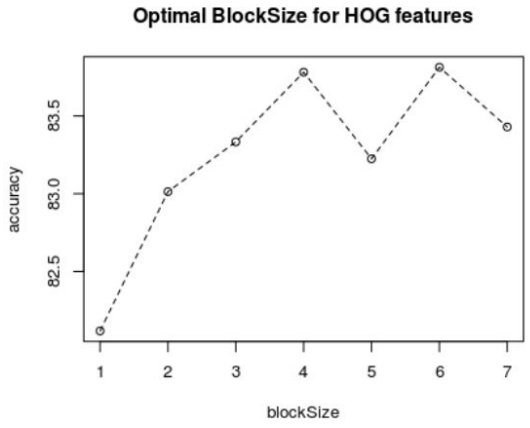
Histograms of Oriented Gradients

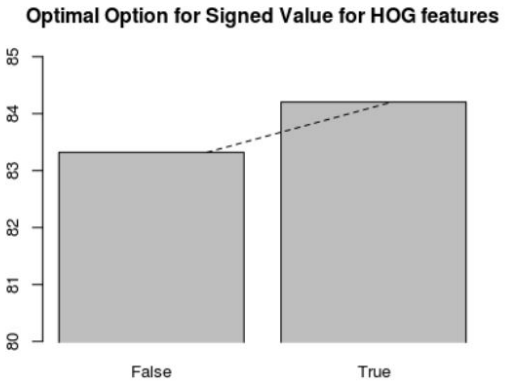
Hem intentat trobar els valors més òptims dels següents paràmetres:

- **Cell Value:** Mida del la finestra de HOG.
- **Nombre de Bins:** Obtenim orientacions més precises si l'augmentem.
- **BlockSize:** Nombre de cel·les per bloc, com més petit és aquest valor, més significància tindran els píxels locals i ajuda a suprimir els canvis d'il·luminació.
- **SignedValue:** Valors d'orientació entre 0º i 180º o entre -180º i 180º. Amb signe obtenim informació sobre el sentit de la transició de les vores de l'histograma.









Crida òptima:

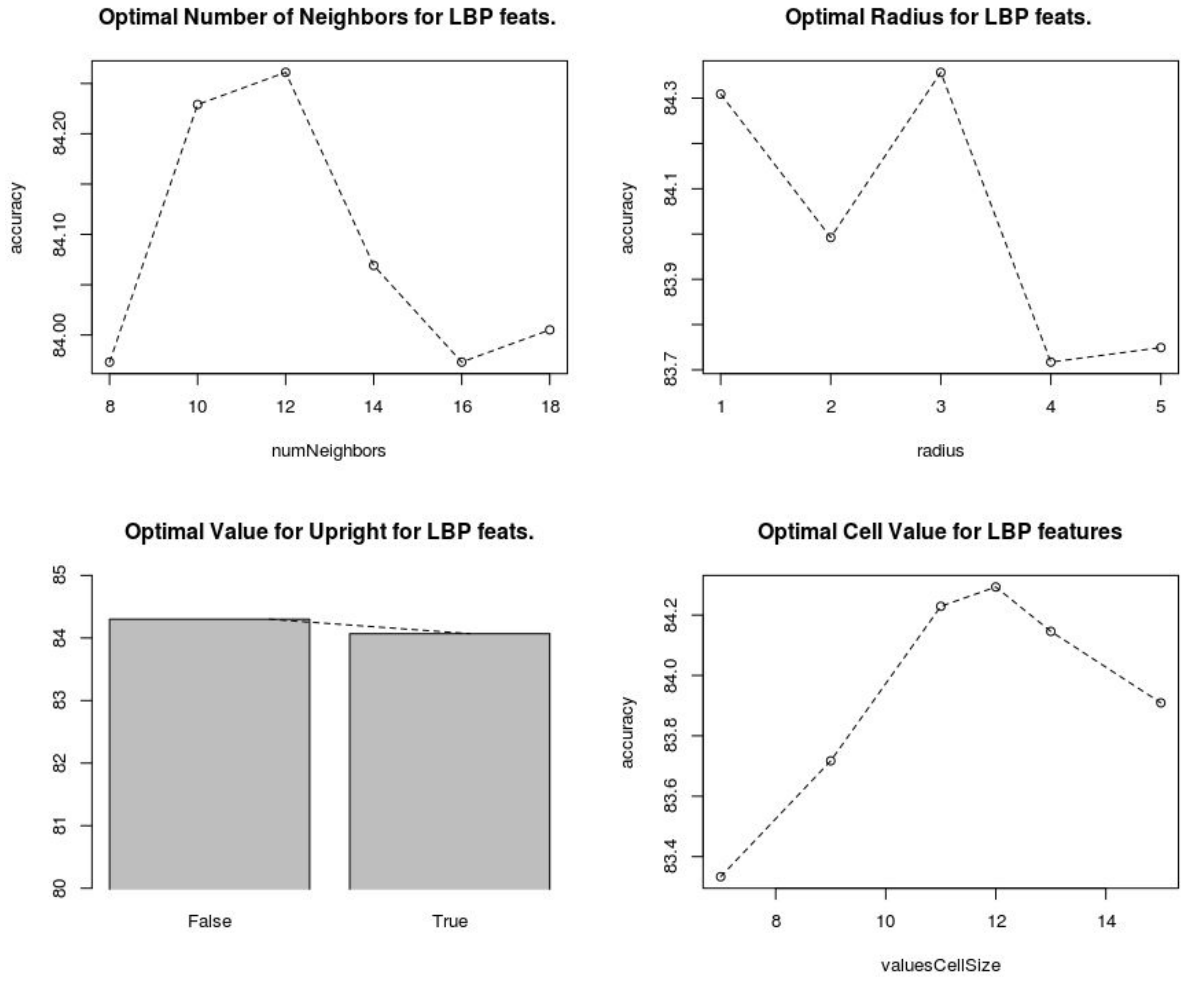
```
featuresHog = extractHOGFeatures(I,'CellSize',[9 9],
                                   'NumBins',10,
                                   'BlockSize',[4 4],
                                   'UseSignedOrientation',true);
```

Local Binary Patterns

Hem intentat trobar els valors més òptims dels següents paràmetres:

- **Cell Value:** Mida del la finestra de LBP.
- **Nombre de veïns:** El nombre de veïns que s'usaran per a computar l'LBP.
- **Radi:** Radi del patró circular per a escollir els veïns que usarà cada píxel.
- **Upright:** Permet fer que les features siguin invariants a la rotació.

Usant HOG amb els millors paràmetres, més LBP:



Crida òptima:

```
featuresLBP = extractLBPFeatures(I, 'NumNeighbors', 12,
                                    'Radius', 3,
                                    'Upright', false,
                                    'CellSize', [12 12]);
```

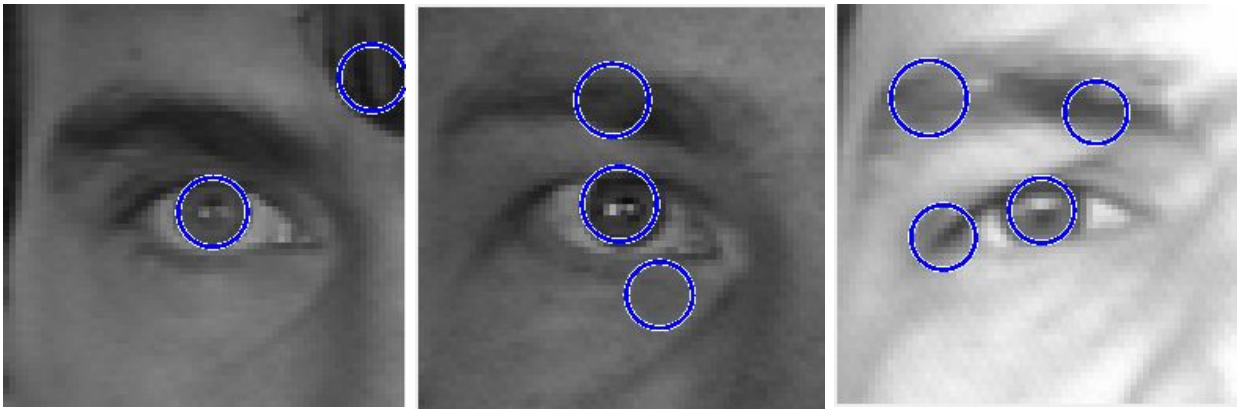
En aquest classificador, la característica de mitjanes de nivell de gris en files i columnes no aportava rellevància per a decidir una classe o l'altre i empitjorava l'accuracy en certa manera. Amb la configuració dels paràmetres d'extracció d'**HOG** + **LBP**, hem arribat a obtenir un model amb un resultat d'encerts de **85.1248**%. Amb matriu de confusió:

Matriu de Confusió	**Observat** **mira**	**no-mira**
Predit **mira**	**328**	**85**
no-mira	**70**	**559**

4. Altres mètodes provats

Transformades de Hough

Veure si pel centre de la imatge es trobava algun cercle que fes referència a la nineta d'un ull (la persona esta mirant directament a càmera). Si es trobava un cercle prop del centre de la imatge, indica que probablement la persona està mirant a càmera.

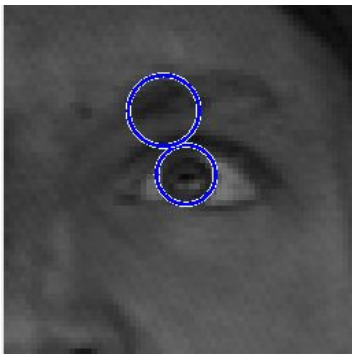


[Funció: `imfindcircles(I, [5 7], 'ObjectPolarity', 'dark', 'Sensitivity', 0.975)`]

Per a ulls que no miren:

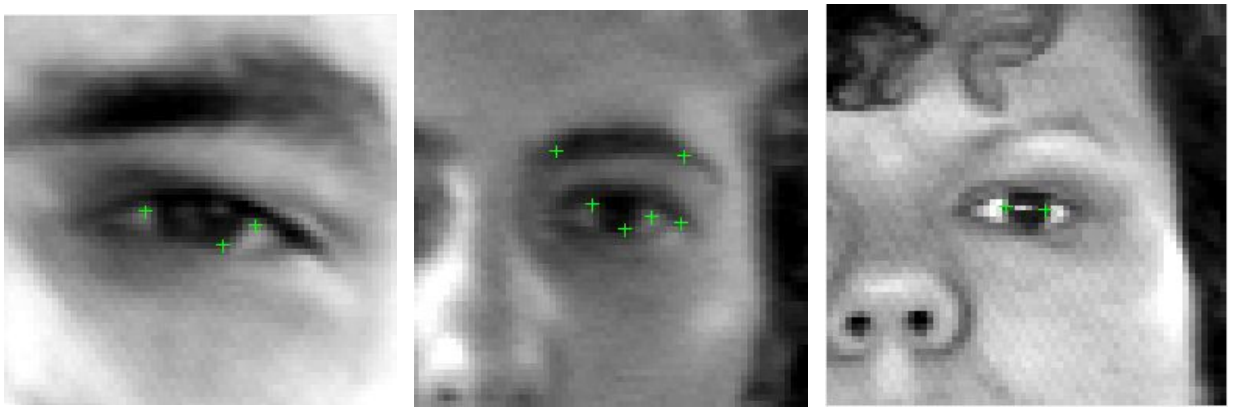
Problema: fallava en bastants imatges i vam haver de posar la sensibilitat de la transformada molt alta (a 97%), amb la qual cosa obteníem cercles soroll i no ens servia per a diferenciar si la nineta tenia forma ovalada (no està mirant a càmera) o forma més circular (mirant a càmera).

En aquesta imatge, l'ull no mira, però la sensibilitat de la transformada de Hough està molt alta i detecta igualment un cercle a la nineta de d'ull.



Keypoints de Harris (corners)

Els corners de Harris defineixen vèrtexs característics de la imatge que podien ajudar a identificar la direcció de la nineta:



[Funció: `corners = detectHarrisFeatures(I,'MinQuality',0.055,'FilterSize',7);`
`[features, valid_corners] = extractFeatures(I, corners);`]

Problema: rebíem un nombre diferent de *keypoints* per a cada imatge amb la qual cosa el nombre de features era variant (el mateix passa amb característiques SIFT, SURF, etc.).

5. Conclusions

Un com acabat el model, l'hem volgut posar a prova amb imatges diferents del dataset. Hem agafat 10 imatges nostres (mirant (1) i no mirant (0) a càmera) per a que el model detectés la nostra mirada.

0 1 0 0 1

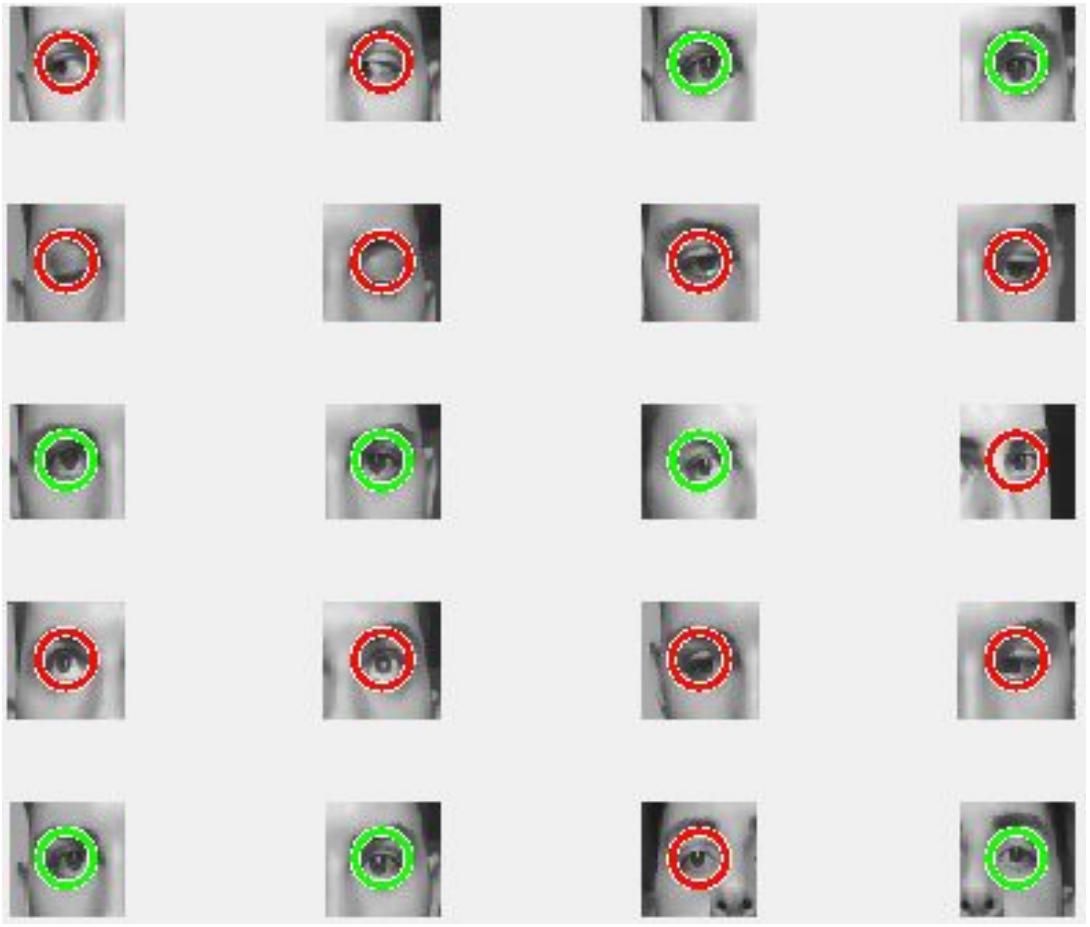
1 0 0 1 1

Primer hem passat una finestra de 64x64 lliscant de 5 en 5 píxels que detectés on es trobaven els ulls (1r classificador).

Veiem que la majoria de punts detectats com a ull són correctes, tot i que tenim també algun fals positiu. D'aquí podem obtenir les coordenades aproximades dels ulls, els quals agafem per separat i determinem quins estan mirant a càmera i quins no (2n classificador).

Per tant el nostre vector d'observacions, com que hem extret dos ulls de cada imatge, és:

[0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1]



Els cercles en verd indiquen que el classificador ha determinat que l'ull mira a càmera, vermell en cas contrari.

scores			Més		Observat
0	1		probable		
0.6200	0.3800		0		0
0.5800	0.4200		0		0
0.4100	0.5900		1		1
0.4800	0.5200		1		1
0.6100	0.3900		0		0
0.5100	0.4900		0		0
0.5500	0.4500		0		0
0.5500	0.4500		0		0
0.4500	0.5500		1		1
0.4800	0.5200	---------->	1		1
0.4300	0.5700		1		1
0.5400	0.4600		**0**	----------	1
0.5700	0.4300		0		0
0.5800	0.4200		0		0
0.5600	0.4400		0		0
0.5600	0.4400		0		0
0.4900	0.5100		1		1
0.4800	0.5200		1		1
0.5900	0.4100		**0**	----------	1
0.3900	0.6100		1		1

Hem obtingut una taxa d'encert del **90**% amb el conjunt reduït de imatges pròpies, la qual és força satisfactòria, amb una matriu de confusió:

Matriu de Confusió	Observat **mira**	Observat **no-mira**
Predit mira	**10**	**2**
Predit no-mira	**0**	**8**

Com a conclusió del projecte, trobem que hem tingut prou bons resultats. Hem après a utilitzar característiques locals i a sintonitzar els paràmetres òptims de les funcions per tal d'abordar un problema de classificació que actualment podria tenir molta aplicació en tecnologies.

En quant a possibles millores, podríem haver provat altres classificadors (SVM, Neural Networks, etc.) per veure si obteníem un millor model, però el TreeBagger (RandomForest) funciona prou bé i és força ràpid, cosa que, amb tantes imatges i features, hem preferit usar principalment. També podríem haver pogut entrenar el classificador amb imatges diferents de les del dataset o aplicar cross-validation per a obtenir un model més precís.

Bibliografia

Es.mathworks.com. (2018). *Extract interest point descriptors - MATLAB extractFeatures - MathWorks United Kingdom*. [online] Available at: https://es.mathworks.com/help/vision/ref/extractfeatures.html [Accessed 14 Jan. 2018].

BISHOP, C. (2016). *PATTERN RECOGNITION AND MACHINE LEARNING*. [S.l.]: SPRINGER-VERLAG NEW YORK.

En.wikipedia.org. (2018). *Histogram of oriented gradients*. [online] Available at: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients [Accessed 14 Jan. 2018].

En.wikipedia.org. (2018). *Local binary patterns*. [online] Available at: https://en.wikipedia.org/wiki/Local_binary_patterns [Accessed 14 Jan. 2018].

Wang X, Han TX, Yan S. *An HOG-LBP human detector with partial occlusion handling.* Conference Paper in Proceedings / IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision. November 2009.

Annex (Codi principal)

```
%%%%%%%%%%%%%%  APRENENTATGE ULL vs NO ULL  %%%%%%%%%%%%%%%%%%%
% script per llegir les imatges d'un carpeta
csv = xlsread('Miram.xlsx');
observations = csv(:,5);
DIR = '/home/gonzalo/FIB/VC/Project/';
imf = dir([DIR '*.pgm']);  % llista d'imatges amb extensio bmp
coordf = dir([DIR '*.eye']);
n = length(imf); % nombre d'imatges en el directori
tSize = 521;
list = [ones([1 n-tSize]) zeros([1 tSize])];
randlist = list(randperm(length(list)));
randlist = randperm(n);

%Treballem amb un 70% (aprox) dels ulls
j = 1; k = 0;
testu = zeros(64,64);
sizeFeat = size(feature_extraction_eyedetection(testu),2);

obsMiraTrain = zeros([2*(n-tSize) 1]);
obsMiraTest = zeros([2*tSize 1]);
oUlls = zeros([2*(n-tSize),sizeFeat]);
oNoUlls = zeros([18*(n-tSize),sizeFeat]);
ulls = uint8(zeros([64,64,2*(n-tSize)]));
for ii = 1 : n-tSize
    i = randlist(ii);
    name = imf(i).name;
    namec = coordf(i). name;
    im = imread(strcat(DIR, name));
    s = size(im);
    l = length(s);
    if l == 3
        im = rgb2gray(im);
    end
    %Extracció de coordenades
    [c1,c2,c3,c4] = textread(strcat(DIR, namec),'%s %s %s %s');
    lx = str2double(cell2mat(c1(2)));
    ly = str2double(cell2mat(c2(2)));
    rx = str2double(cell2mat(c3(2)));
    ry = str2double(cell2mat(c4(2)));

    leftE = [lx-32 ly-32 63 63];
    rightE = [rx-32 ry-32 63 63];
    Il = imcrop(im,leftE);
    Ir = imcrop(im,rightE);

    %Ara toca generar les 18 imatges no-ull
    for index = 1 : 18
        correcte = 0;
      %Comprovem si les coordenades són bones
      while correcte == 0
          correcte = 1;
          coordfila = randi(286-64);
          coordcolumna = randi(384-64);
          %Comprovem que estigui lluny de l'ull dret
          if (coordfila > ry-64) && (coordfila < ry+64)
              if (coordcolumna > rx-64) && (coordcolumna < rx+64)
                  correcte = 0;
              end
          end
          %Comprovem que estigui lluny de l'ull esquerre
          if (coordfila > ly-64) && (coordfila < ly+64)
              if (coordcolumna > lx-64) && (coordcolumna < lx+64)
                  correcte = 0;
              end
          end
      end
```

```
        %Les coordenades son bones un cop aqui
        coordsret = [coordcolumna coordfila 63 63];
        InoUll = imcrop(im,coordsret);
        InoUllr = imresize(InoUll, [64 64]);
        oNoUlls(k+index,:) = feature_extraction_eyedetection(InoUllr);
    end
    %
    Ilr = imresize(Il, [64 64]);
    Irr = imresize(Ir, [64 64]);

    obsMiraTrain(j) = observations(i);
    obsMiraTrain(j+1) = observations(i);
    ulls(:,:,j) = Ilr;
    ulls(:,:,j+1) = Irr;
    oUlls(j,:) = feature_extraction_eyedetection(Ilr);
    oUlls(j+1,:) = feature_extraction_eyedetection(Irr);

    j = j+2;
    k = k+18;
end

ullsuns = ones(2*(n-tSize),1);
noullszeros = zeros(18*(n-tSize),1);
c = [ullsuns ; noullszeros];
o = [oUlls ; oNoUlls];
predictor = TreeBagger(100,o,c);

%TEST%
j = 1;
o2 = zeros([2*tSize,sizeFeat]);
ullstest = uint8(zeros([64,64,2*(tSize)]));
for ii = n-tSize+1:n%n-tSize+1 : n
    i = randlist(ii);
    name = imf(i). name;
    namec = coordf(i). name;
    im = imread(strcat(DIR, name));
    s = size(im);
    l = length(s);
    if l == 3
        im = rgb2gray(im);
    end
    [c1,c2,c3,c4] = textread(strcat(DIR, namec),'%s %s %s %s');
    lx = str2double(cell2mat(c1(2)));
    ly = str2double(cell2mat(c2(2)));
    rx = str2double(cell2mat(c3(2)));
    ry = str2double(cell2mat(c4(2)));

    leftE = [lx-32 ly-32 63 63];
    rightE = [rx-32 ry-32 63 63];
    Il = imcrop(im,leftE);
    Ir = imcrop(im,rightE);
    Ilr = imresize(Il, [64 64]);
    Irr = imresize(Ir, [64 64]);
    obsMiraTest(j) = observations(i);
    obsMiraTest(j+1) = observations(i);
    ullstest(:,:,j) = Ilr;
    ullstest(:,:,j+1) = Irr;
    o2(j,:) = feature_extraction_eyedetection(Ilr);
    o2(j+1,:) = feature_extraction_eyedetection(Irr);
    j = j+2;
end

%Test
[C, scores] = predict(predictor,o2);
predictor.ClassNames

nums = scores(:,1) <= scores (:,2);
%percentatge d'ulls encertats
res = (sum(nums)/size(nums,1))*100
[0 sum(nums == 0); 0 sum(nums==1)]
```

```
%%%%%%%%%  APRENENTATGE MIRA VS NO MIRA %%%%%%%%%%%%%
csv = xlsread('Miram.xlsx');
observations = csv(:,5);
%tSize = 521;

%Creem un vector de uns i zeros per cada ull (el que ve donat es per cada foto)
observtrans = observations';
obscomb = [observtrans; observtrans];
obscomb=obscomb(:);
c = obscomb(1:2*(n-tSize));
%%%%%%%%%%%%%%%%%%%%%
testu = zeros(64,64);
feat = feature_extraction_lookingdetection(testu);
%%%%%%%%
sizeFeat = size(feat,2);
oUllsLook = zeros([2*(n-tSize),sizeFeat]);
%obs1 = zeros([]);
for i = 1 : size(ulls,3)%2*(n-tSize)%2400
   I = ulls(:,:,i);
   oUllsLook(i,:) = feature_extraction_lookingdetection(I);
end

predictorLook = TreeBagger(100,oUllsLook,obsMiraTrain);
oUllsLook2 = zeros([2*tSize,sizeFeat]);
for j = 1 : size(ullstest,3)%tSize*2
   In = ullstest(:,:,j);
   oUllsLook2(j,:) = feature_extraction_lookingdetection(In);
end

[C, scores] = predict(predictorLook,oUllsLook2);
predictorLook.ClassNames

S = scores(:,1)-scores(:,2);
S(S<=0) = -1;
S(S>0) = 0;
S(S == -1) = 1;

kkk = (obsMiraTest == S);
(sum(kkk)/size(kkk,1))*100
[sum((S == 0) .* (obsMiraTest == 0)) sum((S == 0) .* (obsMiraTest == 1)); sum((S == 1) .* (obsMiraTest == 0)) sum((S == 1) .*
(obsMiraTest == 1)) ]

%%%%%%%%%%%%% FUNCTIONS %%%%%%%%%%%%%
function [ features ] = feature_extraction_eyedetection( I )
            I = imadjust(I);
            I = medfilt2(I);
            mcols = mean(I); mrows = mean(I,2);
            featuresMean = [mcols mrows'];
            featuresMean = (featuresMean-min(featuresMean(:))) ./ (max(featuresMean(:)-min(featuresMean(:))));
            featuresHog = extractHOGFeatures(I,'CellSize',[12 12],'NumBins',15,'BlockSize',[2 2], 'UseSignedOrientation',true);
            featuresLBP = extractLBPFeatures(I, 'NumNeighbors', 12,'Radius', 2,'Upright', false,'CellSize', [12 12]);
            features = [featuresLBP featuresHog featuresMean];
end

function [ features ] = feature_extraction_lookingdetection( I )
   I = imadjust(I);
   I = medfilt2(I);
   featuresHog = extractHOGFeatures(I,'CellSize',[9 9],'NumBins',10, 'BlockSize',[4 4], 'UseSignedOrientation',true);
   featuresLBP = extractLBPFeatures(I,'NumNeighbors', 12, 'Radius',3,'Upright',false,'CellSize',[12 12]);
   features = [featuresHog featuresLBP];
end
```