Detector robust de persones mitjançant visió per computador

Memòria projecte

Manel Rosa Martin

Director: Joan Climent Vilaró

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Quadrimestre primavera 2017/2018

Índex

1.-Resum

En aquest projecte hem desenvolupat un sistema robust per a detectar persones dretes de cara a la càmera com seria el cas de un interlocutor. Aquest sistema es basa en la detecció de certes parts del cos que seran fusionades tenint en compte la seva configuració geomètrica per determinar si hi ha una persona en una imatge i on.

Analitzem els possibles costos de desenvolupar el sistema i concloem que es factible mitjançant un desglossament dels costos i recursos. Descrivim el funcionament del sistema i els criteris de disseny d'aquest. Experiments de validació han demostrat que el sistema es de veritat robust amb una precisió del 99% sobre un conjunt de imatges de test. Discutim possibles limitacions del sistema i punts febles. Finalment concloem suggerint possibles millores del sistema.

2.-Introducció i contextualització:

El problema de la detecció de humans es pot definir com donada una imatge localitzar totes les persones en ella i determinar quina regió ocupen en la imatge. La detecció de humans es un problema important per un gran rang de problemes. Com ara sistemes de vídeo vigilància, comptadors de persones, sistemes de frenada autònoms entre altres.

La detecció de humans es un problema de detecció de objectes particularment complicat degut a les característiques de les persones. Detectar persones es un problema difícil degut a la gran variància de la aparença entre individus. Podem trobar persones en moltes posicions diferents, realitzant diferents accions, vestides de manera diferents, des de diferents perspectives...

Una aplicació de la detecció de humans particular son els robots socials. Un robot social es un robot que interactua amb persones i/o altres agents seguint normes socials. Aquests robots poden per exemple ajudar a persones amb mobilitat reduïda [1] o ajudar a desenvolupar habilitats socials a nens amb autisme [2].

També hi han robots socials que poden dur a terme treballs de "coll blanc" o serveis. Com ara recepcionistes o guies de un museu. Aquests robots doncs poden arribar a tindre un clar valor a la societat.

Per a portar a terme aquestes tasques el robots necessiten un sistema per detectar persones amb alta fiabilitat. Aquest projecte pretén dissenyar un sistema que compleixi aquests requisits mitjançant la fusió de diferents tècniques de visió per computador per augmentar la robustesa.

Il·lustració 1 el robot socials necessiten de un sistema de visió per computador robust per dur a terme les seves funcionalitats

2.1.- Estat del art:

En el camp de la detecció de objectes, i en especial humans, es poden trobar diferents famílies de tècniques. Com ara models deformables de parts o models profunds [3].

Els models deformables de parts representen un objecte com a un conjunt de parts i les seves relacions entre geomètriques si. D'aquesta manera es pot tractar casos de oclusió i facilita la detecció amb objectes amb molta variància intra classe com es el cas dels humans. Aquestes parts poden ser seleccionades per un expert o com es més comú inferides de forma automàtica, de vegades de forma implícita. DPM (*Deformable Part Model*) [4] i les seves variants hi son un exemple.

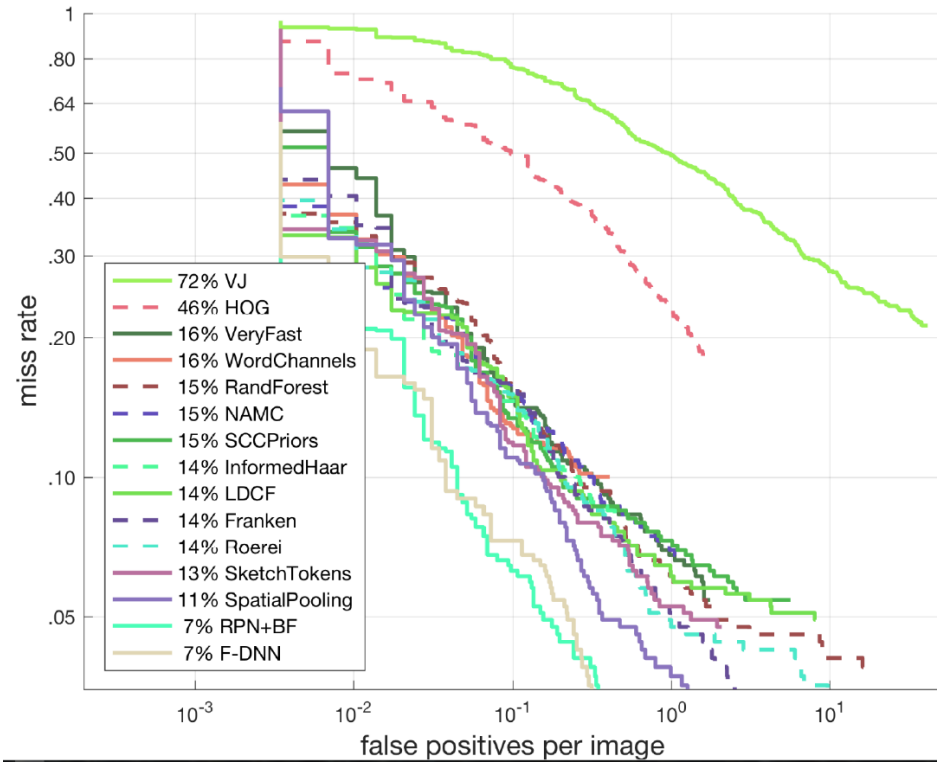
Il·lustració 2 Asimo: segurament el robot social mes conegut

En la actualitat degut al augment de la capacitat de computació els models profunds s'han popularitzat molt. Part del motiu de la seva popularitat es que aquests models infereixen les característiques de les imatges ells mateixos. Es a dir no depenen de "*hand crafted features*" com ara les característiques HOG (*Histogram of Oriented Gradients*) en DPM. En l'instant de redactar aquest document tots els models amb millor rendiment en el *benchmark* PASCAL2012 són profunds [5].Aquests models normalment consisteixen de xarxes neuronals convolucionals profundes. Alguns mètodes usen les xarxes per extreure característiques que després usen en altres mètodes mes clàssics com ara el original R-CNN [6].Mentre que altres com R-FCN son purament convolucionals [7].

Cal destacar que aquestes famílies no son una partició, hi ha models que no pertanyen a aquestes famílies, com per exemple el detector original de Navneet Dalal i Bill Triggs [8]. En general no es pot dir que una família sigui millor que un altre. Tot i que els models profunds dominen PASCAL2012 aquests models son computacionalment costosos a l'hora de entrenar i fer inferència. També s'ha de considerar que per problemes menys complexos altres models més simples en la practica solen rendir millor en part per ser més fàcils de entrenar [3].Per tant no es pot dir que ninguna tècnica ni famílies de tècniques sigui superior a un altre.

A mode de comparació analitzarem el rendiment de un conjunt de tècniques actuals a la detecció de humans. Concretament el dataset de INRIA persons.

Com a resum del rendiment de les tècniques actuals el gràfic (fig 1) mostra el rendiment 13 tècniques actuals i 2 històriques: Viola Jones aplicat a la detecció de humans i el detector de Navneet Dalal i Bill Triggs (HOG). La llegenda conte la tasa de error mitja. El gràfic conte el error respecte al nombre de falsos positius a la imatge a mida que s'augmenta la sensibilitat del detector.



Il·lustració 3 Mostra del rendiment de les tècniques actuals

Il·lustració 4 Mostra de les imatges del dataset INRIA persons

3.- Formulació del problema:

3.1.- Objectius:

El objectiu principal del projecte consisteix en el desenvolupament de un sistema que permeti detectar persones davant de una càmera. Per això dissenyarem un sistema basat en parts que detectarà parts de una persona (ulls, cara, tors...) i integrarà les parts detectades per a determinar si hi ha una persona o no, on i amb quina confiança. Tot i que el projecte esta plantejat en el context de robots socials, aquest sistema ha de ser lo suficientment general com per poder ser usat en un gran nombre de problemes.

El sistema ha de ser robust i lo suficientment eficient com per executar-se en temps real Aquest detector ha de ser robust al detectar persones a una distancia típica de interlocució en posició vertical i de cara a la càmera. No necessàriament a cos complet. El objectiu es detectar un possible interlocutor del robot.

3.2.- Abast:

Primerament tindrem que definir un *dataset* adequat al problema i provar detectors per a cada part del cos. La intenció es reutilitzar tants detectors com sigui possible. A priori la intenció es utilitzar el *Calvin upper body detector* [9] i alguna implantació de un detector de cares Viola Jones. Desenvoluparem el nostre propi detector de ulls.

Una vegada s'ha analitzat el rendiment individual dels detectors s'haurà de explorar diverses estratègies de fusió per a determinar la presencia de una persona. Tindria que ser possible adaptar tècniques de models de parts deformables a aquest problema.

Una vegada s'ha obtingut un detector amb un rendiment de detecció satisfactori es procedirà a optimitzar el seu rendiment computacional, generar documentació tècnica i empaquetar-lo per facilitar la seva utilització.

3.3.- Metodologia i mètode de validació:

Per al desenvolupament del projecte utilitzarem una metodologia àgil. Tot i que estrictament parlant una metodologia àgil requereix de un client i esta pensada per a projectes en equip podem aplicar conceptes. Concretament farem petites iteracions de dues setmanes aproximadament on es validarà el treball fet amb el director que prendrà el rol del client en quant a validació es tracta. D'aquesta manera ens assegurem de detectar problemes aviat per poder solucionar-los més fàcilment i assegurar-nos el assoliment dels objectius del projecte.

4.- Descripció de la solució proposada

La solució que proposem es pot separar en dues grans parts. Un detector de parts i un sistema de fusió. A continuació descrivim la solució proposada.

Primerament es detectaran les parts individuals i despres es fusionaran per determinar si hi ha una persona.

4.1- Detecció de parts

4.1.1.-Detector d'ulls

El detector d'ulls esta basat en "sliding windows" per detectar els ulls a la imatge.

En visió per computador un sliding window en una regió de un mida fixa que es desplaça per la imatge. Per a cada una de les finestres es pot aplicar un classificador per determinar si la finestra correspon a una imatge de una classe d'objecte que ens interessa, ulls en el nostre cas.
Es pot tornar a passar la finestra per la imatge amb diferents mides per tal de detectar objectes a diferents escales. En aquests cas estaríem parlant de un detector en cascada.

Hem entrenat un classificador d'ulls amb el dataset del BioID. Per això s'han extret imatges dels ulls i imatges de "no ulls". Es a dir, trossos extrets de les imatges de manera aleatòria. El objectiu del classificador es determinar si una imatge es d'un ull o no. Aquest problema es molt típic dintre de visió per computador. Consisteix en la extracció de característiques (descriptors) de cada imatge (ull o no ulls) que seran usades per un model classificador per determinar si la imatge es de un ull o no.

Per a entrenar el classificador s'han provat diferents mides de finestra al voltant dels ulls, diferents descriptors: HOG amb diferents paràmetres amb i sense LBP(*Local Binary Pattern*) i diferent models: ensembles (adaboost/gentleboost) amb tree learners i MLP (*Multi layer perceptron*). Tots els híper-paràmetres rellevants dels models s'han optimitzat utilitzant optimització bayesiana.

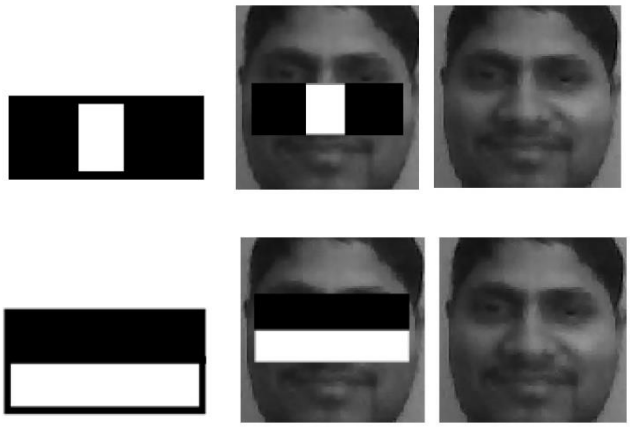
S'ha acabat escollint un classificador amb característiques HOG i un MLP com a model en part per la facilitat de ajustar la sensibilitat del detector.

Per a augmentar el rendiment del sistema només detectarem ulls a les regions on s'han detectat cares.

4.1.2.- Detector de cares

El detector de cares esta basat en el algorisme de Viola Jones.

El algorisme de Viola Jones es el algorisme mes usat per a la detecció de cares. Es basa en la extracció de unes característiques de les imatges. La construcció de un classificador i un detector en cascada.



Exemple de caracteristiques haar-like. La caracterisitca es la suma de els pixels en l'area blanca menys la negra.

Les característiques son les "haar-like features". Son característiques molt simples que corresponen a la suma i resta dels valors dels píxels de diferents àrees rectangulars de la imatge.

les característiques haar-like son adients per a detectar cares ja que les cares tenen certes regularitats que aquestes poden explotar. Cada una de aquestes característiques pot formar un classificador "dèbil" que poden ser combinats amb un classificador adaboost per a generar un classificador robust a base de molts d'aquests classificadors dèbils. Combinant aquest classificador amb una cascada dona com a resultat el clàssic algorisme de Viola Jones.

Amb l'objectiu de detectar mes cares en diferents situacions utilitzarem dos detectors de cares. Un entrenat per a cares de perfil i un altre entrenat per cares mirant a la càmera.

4.1.3.- Detector de torsos

El detector Calvin esta bastat en DPM (Deformable Part Model) mencionat al estat del art. Aquest model es un model de parts deformables multiescala. Usa un formalisme de SVM (*Support Vector Machine*) amb variables latents junt a característiques HOG per al càlcul de filtres. A vegades aquest models s'anomenen en la literatura com a LatSVM. Aquest detector retorna les regions corresponents a la parts superior del cos incloent la cara.

4.2.-Detecció humans i fusió de parts:

La altra part del sistema consisteix en la fusió de les parts per a determinar si hi ha una persona i si n'hi ha on esta a la imatge.

Una vegada s'han detectat les parts individuals a la imatge tenim que agrupar cada una de elles en un grup que correspongui a un possible humà. Per això explotem el fet de que les nostres parts cauen una dintre de l'altra. Es a dir, els ulls estan dintre la cara i la cara dintre el tors. Les parts son agrupades de la següent manera:

1 .-Es selecciona la detecció mes gran en àrea que no formi part d'un grup i s'afegeix a un grup nou.

2.-Totes les parts les quals estigui el 90% del àrea estigui dintre la regió d'aquesta part passen a formar part del grup.

3.- Es repeteix 1 fins que totes les deteccions formin part de un grup.

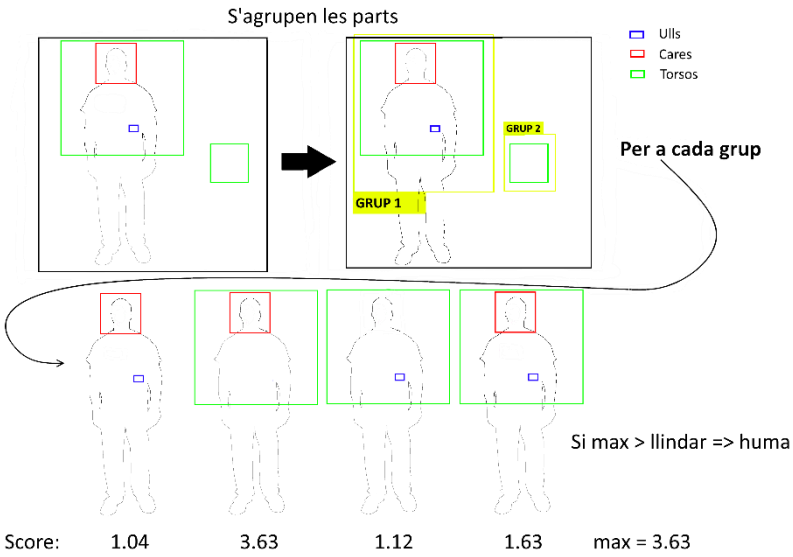
D'aquesta manera podem agrupar les parts de dos humans encara que estiguin molt junts dintre la imatge ja que les parts de les dues persones tindran un overlap petit, amb la excepció de uns pocs casos patològics. Altres algorismes de clustering més típics que hem provat fallen i tendeixen a classificar dos humans junts com a un únic candidat. Aquest simple algorisme funciona molt millor que altres algorismes mes complexos.

Una vegada tenim tots els grups de parts tenim que determinar quins son humans i quins no. Per això calcularem un score per a cada grup seguin una estratègia basada en el score boosting descrit per Mikolajczyk K., Schmid C. et. al [13] amb algunes modificacions que ara descriurem.

Per extreure el score de un grup de parts primerament fem el score boosting descrit a [13] secció 4 per a totes elles. Aquest boosting augmenta el score de les parts, possiblement mes enllà de 1, en funció de la seva posició relativa i el seu score a priori utilitzant uns models gaussians que hem entrenat a partir de les posicions de les parts a les imatges. Per a calcular el score del grup calculem la mediana del score de les parts. La estratègia de boosting la descrivim amb mes detall a un apartat a continuació.

Si apliquéssim aquesta tècnica directament sobre els grups candidats veuríem que falsos positius dels detectors tendeixen a baixar el score del clúster, ja que solen caure en posicions incorrectes que son penalitzades per el score boosting, causant múltiples falsos negatius. Per a solucionar aquest problema i fer la nostra estratègia robusta a falsos positius avaluàrem totes les combinacions de parts dintre el clúster seguint la estratègia descrita anteriorment i ens quedarem amb el score mes gran com a score del grup. Si aquest score es superior a un llindar llavors classifiquem el grup com a humà.

D'aquesta manera ens assegurarem que el score sigui igual o superior a el score sense falsos positius, ja que en algun moment al avaluar totes les combinacions avaluàrem nomes les parts correctes sense falsos positius, si es que n'hi han al grup. Aquesta estratègia no es problemàtica ja que sol haver poques parts dintre els grups.



Il·lustració 7 Diagrama del sistema de fusió amb falsos positius: Després del agrupament de les parts per a cada grup s'avaluen totes les combinacions de parts i si el score màxim es superior a un llindar es classifica com a humà

Aquesta estratègia ens permet augmentar el llindar de classificació de les parts individuals ja que es particularment resistent a falsos positius. Com a resultat, el nostre sistema gaudeix de un recall significativament superior a si no avaluéssim totes les combinacions sense una penalització de precisió.

Il·lustració 8 Mostra de deteccions

En la il·lustració 8 podem veure els grups, les parts individuals i la confiança (score del grup). El àrea del grup es la envolupant de totes les parts, que normalment coincideix amb el tors. Podem veure com s'han agrupat correctament les parts tot i que hi tenen cert overlap, en especial en la primera imatge. En general es pot apreciar com una detecció amb mes parts te una confiança superior.

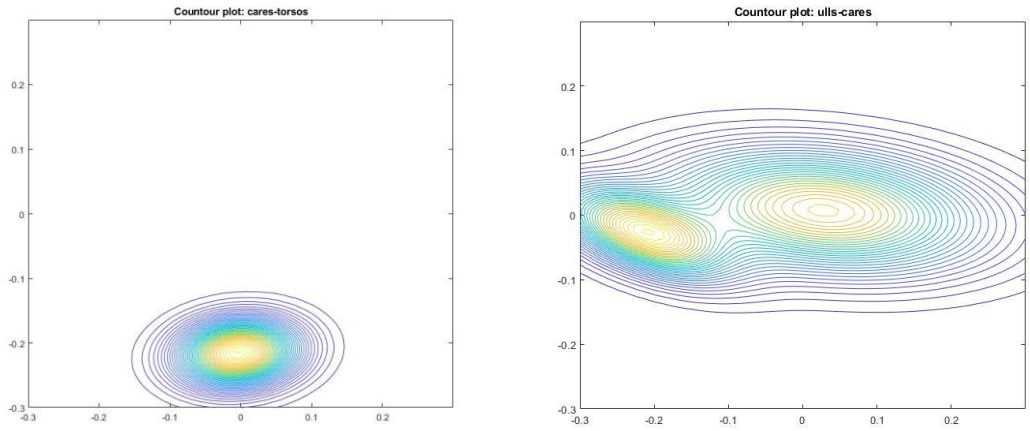
4.3.-Score boosting:

Com hem mencionat el score boosting es basa en el algorisme descrit per Mikolajczyk K., Schmid C. et. al [13]. Farem ara un petit resum mes detallat de la estratègia usada.

El algorisme augmenta la confiança de les parts en funció de la seva relació geomètrica respecte a les parts dintre del mateix grup. Per això seguint el paper [13] s'ordenen les parts per score a priori ,es a dir, per ordre de confiança de detecció. De major a menor. Comencem agafant les dues primeres parts i se li suma a la seva confiança: $\delta\left(\frac{(x_{p1}-x_{p2})}{\sqrt{A_{pm}}}\right) * S_{p2}$ on A_{pm} es l'àrea de la regió mes gran de les dues parts. x_{p1}, x_{p2} son els centroides de la primera i segona part, S_{pi} es la confiança de la part iesima i δ es un model que relaciona les posicions entre el tipus de la part 1 i 2.

Aquesta estratègia es repeteix de parell en parell fins a arribar al parell amb menys confiança de detecció. Es a dir es fa el boosting de la part 1 amb la 2, 2 amb la 3, etc...

En el nostre cas δ es una distribució gaussiana multivariada. En el cas que una de les dues parts sigui un ull serà una mixture of gaussians amb 2 gaussianes. Per tant hi ha una δ per a cada parell de tipus de parts possibles. Es a dir una δ que relaciona ulls i cares, ulls i torsos, ulls i ulls... amb la excepció de quan els dos tipus son torsos o cares. Ja que una persona només te una cara i un tors. Conseqüentment, la probabilitat que hi tingui 2 es 0. Els paràmetres de les distribucions δ son calculats amb el clàssic algorisme EM.



Il·lustració 9 Contour plot de la probabilitat cumulativa de les distribucions δ cares-torsos (esquerra) i ulls-cares (dreta)

Per acabar a mode il·lustratiu observem el contour plot de un parell de distribucions. Com es pot apreciar al plot de les distribucions les cares estan per sobre els torsos. I els ulls estan a la dreta o esquerra del centroide de la cara en funció de si son l'ull esquerra o dret com es d'esperar.

5.-Implementació

Aquest capítol pretén ser una guia tècnica al projecte. Com a tal es comenta el detalls de implementació de les parts critiques del projecte en profunditat per que aquest pugi arribar a ser modificat o estes per un tercer. No pretén ser una guia d'usuari. Per a una guia d'us llegir la secció "us del sistema"

El sistema esta principalment implementat en MATLAB, amb alguns petits trossos en OpenCV/c++. La motivació per utilitzar MATLAB ve principalment donada per la intenció de utilitzar el Calvin upper body detector [9] ja que aquest esta escrit en MATLAB. MATLAB també te el avantatge de que fer prototips en visió per computador es més ràpid, ja que conte tot un paquet de suport per el desenvolupament de aplicacions de visió per computador. Dit això te el inconvenient de ser propietari amb llicencia de pagament i es difícil de optimitzar codi crític per el rendiment. En gran part degut a la manca de control sobre la memòria.

5.1.- El dataset

Per a dur a terme aquest projecte tindrem que desenvolupar i dissenyar diferents models de classificació. Per això com a tot problema de machine learning es necessari definir un dataset sobre el que entrenar i validar els models.

El dataset ha de ser representatiu del nostre problema. Es a dir ha de ser compost de imatges de persones dretes de cara a la camara en situacions variades. Amb una resolució lo suficientment elevada per a poder detectar parts petites com ara els ulls.

Després de comparar els datasets disponibles s'ha definit un dataset a partir de dos datasets publics. El bbc pose [11] dels VGG datasets I el buffy stickmen [12]. S'han etiquetat manualment els ulls, cares I torsos de les imatges del dataset de forma manual amb una petita aplicació desenvolupada per a la tasca.

Aquest dataset conte persones en una gran varietat de posicions i en una gran varietat de ambients i diferents situacions de il·luminació. Com a tal considerem que es representatiu de la realitat del nostre problema.

Il·lustració 5 Exemples de imatges del dataset

5.2- Implementació del detector de parts

Com hem mencionat anteriorment. El projecte es pot separar en dues parts. Un detector de parts i una part de fusió. A continuació descriurem el detector de parts.

El detector de parts donada una imatge retorna les regions on es troben les possibles cares, torsos i ulls junt amb la confiança de cada detecció.

Cada detector de parts te la seva funció que detecta en una imatge la part en qüestió. Ara descrivim la implementació de cada detector individual:

5.2.1.- Detector ulls

El detector d'ulls esta implementat en detecteyes.m:

La funció detecteyes es composa de:

```
%wr tamany de finestra
%net red de classificacio
%x es l'imatge
function [rois,cluster_means,cluster_scores] = detecteyes(x,net,wr)
<<Calcul de vector caracteristiques>>
<<Classificacio>>
<<Agrupament deteccions>>
```

Per eficiència, calculem totes les característiques de les finestres i les posem a un vector.

```
<<Calcul de vector caracteristiques>>=
winy = 1:(wr(2));
winx = 1:(wr(1));

%Calculem el nombre de finestres
s = size(x);
ny = idivide(s(1)-wr(2)-1,int32(stridey));
nx = idivide(s(2)-wr(1)-1,int32(stridex));

winfeat = zeros([1251,ny,nx]);
for ypos = (1:(ny))
     for xpos = (1:(nx))
         currenty = winy + double(stridey*(ypos-1));
         currentx = winx + double(stridex*(xpos-1));
         %extraiem la regio de la finestra
         i = x(currenty,currentx);
         f = featurevector(i)';
         %i posem guardem les caracteristiques al vector
         winfeat(:,ypos,xpos) = f;
     end
end
```

Aquesta part es una de les menys eficients del sistema. Degut a la incapacitat de matlab de fer operacions “in-place”. Conseqüentment cada finestra es una copia de la regió de la imatge.

Seguidament classifiquem totes les finestres d’un cop, passant les característiques per la xarxa net. Llavors comprovem si cada finestra per si passa d’un cert llindar. Si es així, aquesta finestra passa a considerar-se un ull i la seva posició e índex es guarden per més endavant.

```
<<Classificacio>>=
%Classificar cada finestra amb el MLP
yp = net(winfeat);
clust = [];
indxclust = [];

for ypos = (0:(ny-1))
     for xpos = (0:(nx-1))
         %Si la finestra te un score major que un llindar
         if(yp(ypos*nx + xpos +1) > eyeClassificationThreshold)
            %guardar el centroide de la finesta al vector clust
             clust = [clust;xpos*stridex + wr(1)/2,…
                          ypos*stridey + wr(2)/2];
             %i el index de la finestra
             indxclust = [indxclust (ypos*nx + xpos +1)];
         end
     end
end
```

Finalment ja que sol haver moltes deteccions positives al voltant dels ulls es necessari agrupar les deteccions en una única. Per això fem clustering de els punts detectats. Calculem el score del clúster com a el màxim score de els punts corresponents al clúster.

Calculem el punt mitja de cada clúster i la finestra al seu voltant. Aquest pesarà a ser la detecció final del ull.

```
<<Agrupament deteccions>>=
%Cluster basat en densitat,trobar indexos del cluster
indx = DBSCANscpt(double(clust),15,1);
indx = indx';
cluster = unique(indx);
cluster_means = [];
cluster_scores = [];
rois = [];

%calcular el punt mitja de cada cluster
%i el score maxim
for i = cluster
    points = clust(indx == i,:);
    m = mean(points,1);
    scores = yp(indxclust(indx == i));
    score = max(scores);
    cluster_scores = [cluster_scores; score];
    cluster_means = [cluster_means;m];
    rois = [rois; (clust - win) (win*2+1)];
end
```

La xarxa de classificació net s'ha entrenat a partir de les imatges dels ulls del dataset BioID. S'ha extret trossos aleatoris de les imatges per als exemples negatius.

Per entrenar una xarxa existeix la funció [loss,net] = entrenarred(nfirst,nsec,feat,y) que entrena una xarxa amb dues capes ocultes amb nfirst i nsec neurones cada una. On cada fila de feat representa el vector de característiques de una imatge i y es un vector amb 0 al índex i si la fila de feat representa una imatge no ulls i 1 per ulls. La funció retorna el resultat de validació en un 30% de les imatges escollides aleatòriament.

```
function [loss,net] = entrenarred(nfirst,nsec,feat,y)
x = feat';
t = squeeze(y);
trainFcn = 'trainscg';

hiddenLayerSize = [nfirst nsec ];
net = patternnet(hiddenLayerSize, trainFcn);

net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 10/100;

[net,tr] = train(net,x,t,'useGPU','no');
loss = tr.best_vperf;
```

Llavors els paràmetres nfirst i nsec poden ser optimitzats bayesianament amb el script optimize.m.

```
nfirst = optimizableVariable('nfirst',[100,3000],'Type','integer');
nsecond = optimizableVariable('nsecond',[100,2000],'Type','integer');

fun = @(x)entrenarred(x.nfirst,x.nsecond,feat,y')
results = bayesopt(fun,[nfirst nsecond],'MaxObjectiveEvaluations',60)
```

La xarxa net que usem per el detector de ulls es el resultat d'aquesta optimització bayesiana.

Existeix un script trainensemble.m que entrena un model adaboost/gentleboost. Aquest script es va usar per comparar el rendiment de diferents models de classificació.

5.2.2.- Detector cares:

La funció del detector esta implementada en detectFaces.m.

```
function [faces,scores] = detectFaces(I)
<<trucadaDetectorOpenCV>>
<<mapejatScore>>
```

El detector de cares es una implementació de OpenCV. Com a tal, es truca a una funció foranea escrita en c++ que retorna les cares detectades i els scores.

```
<<trucadaDetectorOpenCV>>=
[faces,scores] = detectFacesOpenCV(im2uint8(I));
```

Després de les deteccions mapejem el score cap al rang [0,1]

```
<<mapejatScore>>=
%logit score transform
scores = arrayfun(@(x) 1/(1+exp(-(x*0.95 - 100))),scores);
```

La funció forana detectFacesOpenCV esta implementada en detectFacesOpenCV.cpp

Aquesta part ha tingut molts problemes tècnics. Existeix una interfície nativa de MATLAB a OpenCV. Aquesta interfície consisteix en uns binaris precompilats de OpenCV 3.1 i una sèrie de funcions de conversió de dades. El problema consisteix en que la versió precompilada de matlab es molt antiga i conte bugs que fan impossible extreure la confiança de detecció.

Conseqüentment s'ha tingut que usar la interfície c++ i llincar manualment amb uns binaris d'una versió mes moderna de OpenCV. A mes de usar una altra llibreria per a la conversió de tipus. Aquesta es la rao per la qual el sistema no funcionara directament sota Linux ja que s'ha de llincar amb un binari Linux de OpenCV. El script buildmex.m es el que llinca i compila el arxiu detectFacesOpenCV.cpp.

```
detectFacesOpenCV.cpp =

<<Includes>>

void mexFunction(int nlhs, mxArray *plhs[],…
                 int nrhs, const mxArray *prhs[])
{

  <<Conversió de tipus matlab-c++>>
  <<Carrega Models OpenCV>>
  <<Detecció cares>>
  <<Eliminació cares duplicades>>
  << Conversió de tipus c++-matlab>>
}
```

mexFunction es el punt d'entrada en totes les trucades foranies de matlab cap a c++. Els paràmetres son el nombre de paràmetres de entrada i sortida de la funció de matlab, a mes dels mxArrays corresponent als paràmetres. mxArray es un tipus que representa una matriu a matlab. Per això primerament s'ha de convertir cap a un tipus de OpenCV. Per a la conversió de tipus usem la llibreria mexopencv.

```
<<Conversió de tipus matlab-c++>>=
MxArray arr(prhs[0]);
cv::Mat img = arr.toMat();
```

Seguidament carreguem els models pre-entrenats de OpenCV:

```
<<Carrega Models OpenCV>>=
    cv::CascadeClassifier face_cascade;
    if (!face_cascade.load(face_cascade_name)) {
        printf("--(!)Error loading\n");
        return;
    };

    cv::CascadeClassifier sideface_cascade;
    if (!sideface_cascade.load(faceprofile_cascade_name)) {
        printf("--(!)Error loading\n");
        return;
    };
```

I procedim a la detecció de les cares. Aquest detector es una cascada estàndard de viola Jones.

```
<<Detecció cares>>=
    std::vector<cv::Rect> faces;
    std::vector<int> reject_levels;
    std::vector<double> level_weights;

    face_cascade.detectMultiScale(img,faces,reject_levels,
                                  level_weights,1.1,3,0,
                                  cv::Size(),
                                  cv::Size(), true);


    std::vector<cv::Rect> sidefaces;
    std::vector<int> sidereject_levels;
    std::vector<double> sidelevel_weights;
    sideface_cascade.detectMultiScale(img,sidefaces,sidereject_levels,
                                      sidelevel_weights,1.1,3,0,
                                      cv::Size(),
                                      cv::Size(), true);
```

El detector de cares en perfil i normal solen detectar la mateixa cara en moltes situacions, per això fa falta una mica de post-processat. Iterarem per a totes les cares de el detector de cares en perfil i si no hi ha overlap amb ninguna cara afegim la cara detectada al vector que retornarem,

<<Eliminació cares duplicades>>=

```
for(int i = 0; i < sidefaces.size();i++){
        bool repeated = false;
        for(int k = 0; k < faces.size();k++){
            if(overlap(sidefaces[i],faces[k]) > 0){
                repeated = true;
                break;
            }
        }
        if(!repeated){
            faces.push_back(sidefaces[i]);
            level_weights.push_back(sidelevel_weights[i]);
        }
    }
```

Finalment retornem el vector de regions i scores. Per això tenim que tornar a convertir els tipus de dades.

<< Conversió de tipus c++-matlab>>=

```
    MxArray arrfaces(faces);
    //Primer parametre regions
    if(nlhs >= 1){
        plhs[0] = arrfaces;
    }
    //Segon parametre scores
    if(nlhs >= 2){
        plhs[1] = mxCreateNumericMatrix( 1,
                                         level_weights.size(),
                                         mxDOUBLE_CLASS,
                                         mxREAL );
        memcpy( mxGetData(plhs[1]),
                &level_weights[0],
                level_weights.size()*sizeof(double) );
    }
```

5.2.3.- Detector torsos.

La detecció de torsos es trivial ja que el calvin esta implementat en Matlab. Nomes fa falta carregar els models i trucar a DetectStillImage().

Per detectar torsos:

```
load('detenv.mat');
pffmodel = load('pff_model_upperbody_final.mat');
upperBodies = DetectStillImage(I,pffmodel,[],det_pars,0);
```

5.2.4.-Funció detectParts

La funció principal del detector de parts es [rois,classes,scores]= detectParts(I).

Donada una imatge, aquesta funció retorna les regions de les parts(rectangles de la imatge on hi ha una part). Classes,cell array de strings indicant si cada part es una cara, tors o ull. I la confiança de les deteccions.

detectParts esta implementat a detectParts.m de la seguent manera:

```
%rois: regions detectades
%classes: tipus de deteccio. ulls,cares o torsos
%scores: confinaça deteccio
function [rois,classes,scores] = detectParts(I)
<<detectarTorsos>>
<<detectarCares>>
<<detectarUlls>>
<<agruparDeteccions>>
```

Primerament, per millorar el rendiment els torsos es detecten en paral·lel. Per això truquem asincroanment la funció calvinDetect.

```
<<detectarTorsos>>=
F = parfeval(@calvinDetect,3,I,pffmodel,det_pars);
```

calvinDetect esta implementat en calvinDetect.m. Aquesta funció retorna les regions de torsos detectades amb les seves confiances junt a un cell array amb les etiquetes de classe, un string que identifica la detecció com a torso.

```
function [upperBodies,upperscores,upperlabel] = calvinDetect(I,pffmodel,det_pars)
<<detectarTorsos>>
<<ExtreureConfiaçes i etiquetes>>
<<MapejarScore>>
```

Per a detectar els torsos es truca a la funció del detector calvin.

```
<<DetectarTorsos>>=
upperBodies = DetectStillImage(I,pffmodel,[],det_pars,0);
```

A continuació es genera el vector de scores i regions, junt a les classes.

```
<<ExtreureConfiaçes i etiquetes>>=
upperscores = [];
upperlabel = cell([1,size(upperBodies,1)]);


for i=1:size(upperBodies,1)
    upbd = upperBodies(i,:);
    upperBodies(i,:) = [upbd(1:2) (upbd(3:4)-upbd(1:2)+1) upbd(5)];
    upperlabel{i} = 'torsos';
    upperscores = [upperscores upbd(5)];
end
```

Finalment es pasen els scores al rang [0,1] amb un *<<logit score transform>>*

```
<<MapejarScore>>=
upperscores = arrayfun(@(x) 1/(1+exp(-(x+2))),upperscores); %logit score transform
```

Per a detectar cares es truca a la funció del detector de cares detectFaces(I). També es genera un vector d'etiquetes.

```
<<detectarCares>>
[faces,facescores] = detectFaces(I);
facelabel = cell([1,size(faces,1)]);
facelabel(:) = {'cares'};
```

Per a detectar els ulls, s'extreu la imatge corresponent cada regió corresponent a les cares i s'executa detectEyes() a aquesta.

```
<<detectarUlls>>=
eyescores = [];
eyes = [];
for i=1:size(faces,1)
     face = faces(i,:);
     facei = imcrop(I,face);
     [eyesf,~,eyescoresf] = detecteyes(facei,net,[33 35]);
     if size(eyesf,1) > 0
        eyesf(:,1:2) = eyesf(:,1:2) + face(1:2);
        eyes = [eyes;eyesf ];
        eyescores = [eyescores, eyescoresf];
     end
end

eyelabel= cell([1,size(faces,1)]);
eyelabel(:) = {'cares'};
```

Ja que hem executat calvinDetect asincronament tenim que esperar a que finalitzi i extreure la sortida la funció.

Finalment, es concatenen totes les deteccions en uns únics vectors.

```
<<AgruparDeteccions>>=

%Esperar a calvinDetect
wait(F);
[~,upperBodies,upperscores,upperlabel] = fetchNext(F);
%Concatenar deteccions
rois = [faces;upperBodies;eyes];
classes = [facelabel upperlabel eyelabel];
scores = [facescores upperscores eyescores];
```

5.3.- Implementació fusió

La funció principal de tot el sistema es detectHuman(img). Aquesta funció donada una imatge retorna les regions on s'ha detectat un humà a mes de amb quina confiança, junt amb la imatge original amb les deteccions marcades.

Aquesta funció esta implementada en detectHuman.m de la següent manera:

```
%roiout: regions detectades
%scores: confiança deteccio
%iout: imatge amb les deteccions marcades
function [roiout,scores,iout] = detectHuman(img)
<<detectarParts>>
<<agruparParts>>
<<detectarHumans>>
```

Primerament es detecten totes les parts. Per això es truca da la funció detectParts(). També es calculen els centroides de les regions.

```
<<detectarParts>>=
[rois,classes,scores] = detectParts(img);
%Calcular els centroides de les rois
points = [];
for i= 1:size(rois,1)
    points = [points;centr(rois(i,:))];
end
```

Seguidament s'agrupen les parts amb el algorisme d'agrupament descrit anteriorment en la secció de descripció de la solució.

```
<<agruparParts>>=
%Odenar les rois per areas
areas = arrayfun(@(indx) prod(rois(indx,3:4)),1:size(rois,1));
[areas,indx] = sort(areas,'descend');

rois = rois(indx,:);
classes = classes(indx);
scores = scores(indx);

%Inicialitzem variables
clusterindex = zeros([1,size(rois,1)]);
cluster = [];

found = find(clusterindex == 0);
%mentres hi hagi rois sense cluster asignat
while size(found,2) > 0
    %Generem un nou id de cluster
    clusterid = size(cluster,2) + 1;
    cluster = [cluster,clusterid];

    %Trobem la roi mes gran sense cluster
    biggestRoi = rois(found(1),:);
    clusterindex(found(1)) = clusterid;

    %I totes les rois dintre d'aquesta
    inroi = inrect(rois(found,:),biggestRoi);

    %Totes les rois que estiguin dintre de
    %la mes gran son part del mateix cluster
    clusterindex(found(inroi)) = clusterid;
    found = find(clusterindex == 0);
end
```

Finalment es classifica cada clúster com a humà o no seguint la estratègia descrita en la secció on hem descrit la solució proposada.

Per això amb les parts que formen cada clúster generem totes les combinacions de entre 2 i 4 parts. Generant el boosted score de cada combinació i quedant-nos amb el mes gran.

```
<<detectarHumans>>=
%Per a cada cluster
for k = cluster
    %extreure els centroides de les parts del cluster
    p = points(clusterindex == k,:);
    %els scores
    clusterScores = scores(clusterindex == k);
    %les classes de les parts
    clusterclasses = classes(clusterindex == k);
    %i les regions de les parts
    crois = rois(clusterindex == k,:);

    maximumClusterScore = 0;

    for nparts = 2:4
        %Generar la matriu de combinacions de totes les parts del cluster
        %de 2 a com a molt 4 parts
        perms = combnk(1:size(p,1),min([nparts,size(p,1)]));
        permroi = [];

        %per a tota permutacio de parts
        for iperm = 1:(size(perms,1))
          <<Fer score boosting>>
           <<calcular score cluster>>
        end
    end
     <<classificar>>
end
```

Per a fer score boosting iterem de par en par per les parts de la permutació. Fent boosting d'acord amb la estratègia descrita anteriorment. Per això usem els models adequats per a el parell de classes.

```
<<Fer score boosting>>=
%index de les parts de la permutacio
perm = perms(iperm,:);
maxrois = crois(perm,:);
%fer score boosting

%iterar de parell en parell de parts
for i = 1:(size(perm,2)-1)
      %rois de la permutacio
      permroi = crois(perm,:);

      %centroides de les dues parts
      ap = p(perm(i),:);
      anp = p(perm(i+1),:);

      %classes i scores de les dues parts
      ac = clusterclasses(perm(i));
      anc = clusterclasses(perm(i+1));
      asc = clusterScores(perm(i));
      ansc = clusterScores(perm(i+1));

      %1 / sqrt(Maxima area de les parts)
      scalefactor = 1 / sqrt(max(prod(permroi(:,3:4)')));

      %No fer score boosting si hi ha mes de 1 torso o cara
      if strcmp(ac,anc) ~= 1 || strcmp(ac,'ulls') == 1
            boostedScore = ansc +
            (1 - cdf(models{classindex(ac),classindex(anc)},
            (anp - ap)*scalefactor))*asc;

            clusterScores(i+1) = boostedScore;
      end
end
```

Després de fer el boosting calcuem els score que es la mitja del score de les parts i, si es el mes gran fins ara actualitzem la variable maximumScoreCluster.

```
<<calcular score cluster>>=
score = median(clusterScores);
if score > maximumClusterScore
      maximumClusterScore = score;
      maxrois = permroi;
end
```

Finalment una vegada calculat es score del clúster es qüestió de classificar com a humà el clúster si aquest passa de un cert valor llindar.

Si es classifica com a humà. Calculem la regió detectada com a la regió envolupant de totes les regions del clúster. Finalment marquem la detecció a la imatge.

```
<<classificar>>=
if maximumClusterScore > classificationThreshold
        minp = min(maxrois(:,1:2),[],1);
        maxp = max(maxrois(:,1:2) + maxrois(:,3:4),[],1);
        env = [minp (maxp-minp + 1)];
        roiout = [roiout;env];
        conf = [conf,maximumClusterScore];

        iout = insertObjectAnnotation(iout,'rectangle',env,…
                ['Confiança : ' num2str(maximumClusterScore)],…
                 'TextBoxOpacity',0.9,'FontSize',12);
end
```

Per entrenar els models geomètrics de les parts. Hem generat per a tot parell de classes tot els vectors diferencia entre els centroides de les parts dividit per l'arrel quadrada de la àrea de la regió major, tal com es descriu a la secció de score boosting. Seguidament s'ha ajustat un model gaussià amb fitgmdist().

El codi que genera els models geomètrics entre classes es troba a fitmgaussian.m.

5.4.- Us del sistema

El sistema consisteix de un conjunt de fitxers de matlab. Junt a un fitxer .mex[1] i llibreries dll.
Per a usar el sistema es suficient de afegir el directori al path de matlab i executar el script init.m.

Aquest script carrega els models i les variables necessàries.

Llavors es pot trucar [rois,scores] = detectHuman(image) que retorna les regions on s'ha detectat un huma junt amb la seva confiança.

El sistema en la actualitat funciona sota Windows. Per executarse sobre Linux o un altre so s'hauria de compilar els arxiu mex i llincar respecte una binari de OpenCV 3.4 o superior.
Per a compilar el arxiu mex es suficient trucar a buildmex.m.

Existeix un arxiu globalVariables.m que conte una sèrie de paràmetres del sistema que poden ser editats per modificar el rendiment del sistema. Com ara els llindars de classificació.

La part de detecció de parts es continguda dintre de detectParts.m. La part de fusió de parts esta dintre de detectHuman.m.

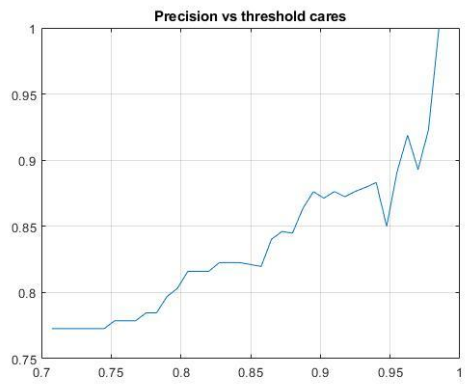
[1] Els arxius mex contenen codi binari compilat amb c++ i son la interfície estàndard de MATLAB amb codi natiu.

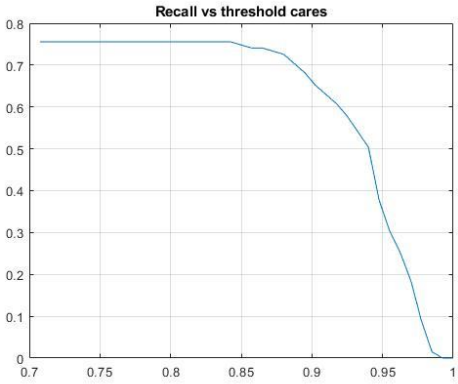
6.-Validació de resultats:

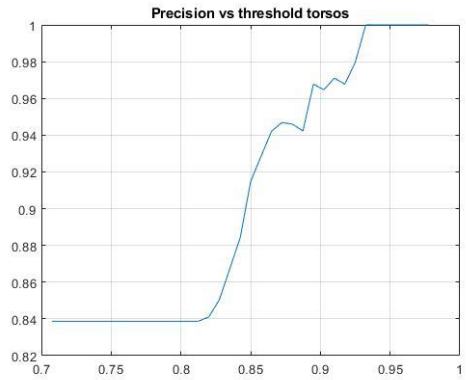
Una vegada integrats els detectors s'ha fet un profiling del rendiment i escollit llindars de classificació adients. Per avaluar el rendiment dels detectors es considera que una detecció es correcta si el overlap de la caixa de detecció es superior a 0.5 amb les regions etiquetades a les imatges. S'adjunta una a taula i gràfics com a resum del rendiment dels detectors.

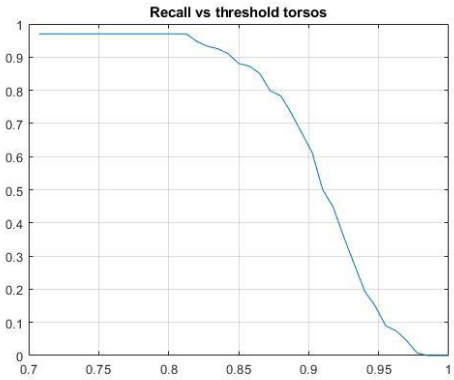
Les confiances dels torsos i cares, ja que utilitzen mètodes ensemble, han sigut mapejades al interval [0,1] amb un *logit score transform.*
La taula inclou la precisió de les deteccions, recall i falsos positius per imatge. A més del llindar de classificació que hem escollit. Es a dir, la confiança a partir la qual es classifica una instancia com a positiva.

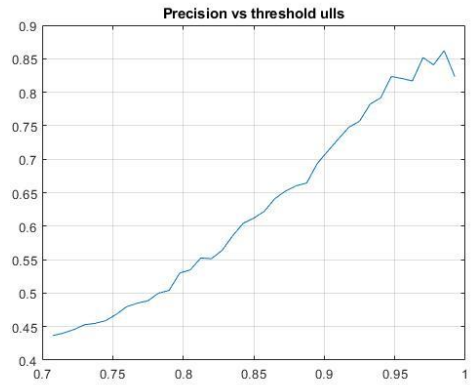
	Precisió	Recall	fppi	Threshold
Cares	0.8462	0.7259	0.2900	> 0.86
Torsos	0.9286	0.8806	0.1600	> 0.86
Ulls	0.7822	0.3196	0.9600	> 0.93

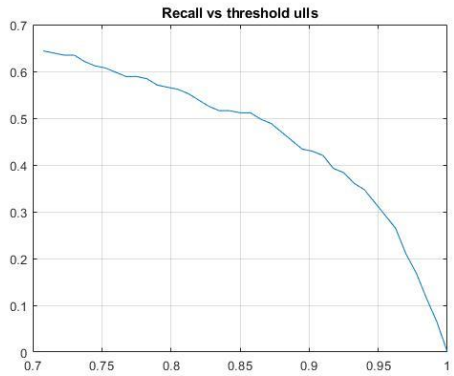










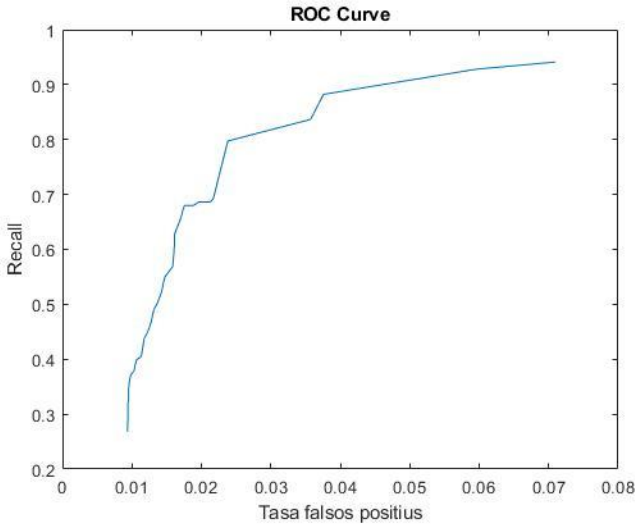


Com podem observar el detector de calvin gaudeix del millor rendiment. Seguit del detector de cares i ulls. Això no es sorprenent ja que el calvin es el detector mes sofisticat i el que detecta la part mes gran. El detector d'ulls pateix en part per la petita mida dels ulls a les imatges. Es molt difícil classificar correctament petites imatges amb baixa resolució especialment en situacions de poca il·luminació.

Per a validar els resultats s'ha definit un dataset de unes 105 imatges de test a partir dels dos mateixos datasets usats per entrenar els models. El dataset de test i entrenament son independents, es a dir, no tenen ninguna imatge en comú.

Hem executat el sistema sobre el dataset de test amb diferents paràmetres. Hem determinat que si l'area de la regió detectada coincideix amb un 80% amb la àrea etiquetada a la imatge com a humà classifiquem com a detecció positiva. A continuació es mostren els resultats. Fa falta mencionar que aquets resultats son fàcilment reproduïbles amb l'script "test.m".

Hem generat la corba ROC (*Receiver Operating Characteristic*) que relaciona el recall amb la tasa de falsos positius a mida que augmentem el llindar de classificació. Podem observar que la tasa de falsos positius en tot cas es molt petita. Això te sentit ja que la precisió dels detectors individuals ronda el 80%. Hem usat aquesta corba per escollir un valor adequat per al llindar de classificació. Hem escollit 1.5 com a llindar que proporciona un recall de 67.9% amb una precisió del 99.05% en el dataset de test. Ja que hem considerat que en el nostre cas preferim un sistema amb més fiabilitat sacrificant una mica de recall.



Il·lustració 10 La corba ROC gaudeix de pocs falsos positius degut a la precisió del detectors individuals

El temps d'execució ronda l'ordre del segon per imatge. La majoria del qual correspon a el detector calvin. Considerem el temps de execució millorable però suficient per a la tasca que ens hem proposat. Possibles millores del sistema passarien per millorar aquest rendiment.

7.- Planificació temporal:

La data límit per el projecte es el 18 de juny de 2018, una setmana abans del començament de les presentacions. La duració aproximada del projecte son 3 mesos.

7.1- Recursos:

Disposem dels següents recursos per al desenvolupament del projecte:

Recursos humans:
Director: valida i guia el projecte.

Recursos hardware:
Per entrenar els models un servidor hp proliant dl160 g6

Per el desenvolupament un ordinador personal estàndard

Recursos software:
MATLAB R2018a. Usat per a tot el projecte sobre una maquina Linux ubuntu 17.10

Per a la organització del projecte usem Microsoft office Project 2013

Per a la organització del codi usem Git

Microsoft Word per la redacció de la documentació.

7.2.- Descripció de tasques i fases del projecte.

Al utilitzar una metodologia àgil el nombre de tasques pot variar a mida que avança el projecte. A l'hora de començar el projecte aquestes son les fases i les seves tasques que podem preveure.

7.2.1.- Fase inicial:

Tasques:

Definició del abast i estudi del estat del art

Planificació del projecte

Els recursos utilitzats per aquestes tasques son:

Recursos humans:

El director guia la definició del projecte i el estat del art

Recursos hardware:

Ordinador personal on es redactarà la documentació

Recursos software:

Microsoft office Project 2013 per planificar el projecte

Microsoft Word per redactar els documents

7.2.2.- Desenvolupament:

Es la fase en que una vegada el projecte esta definit aquest es posa en desenvolupament

Tasques:

-Obtenció del *dataset*.

Per obtenir el *dataset* fa falta veure quins existeixen en la actualitat i quins d'ells es poden adaptar al problema. També inclou etiquetar les imatges amb les parts corresponents per el entrenament del detector. I possiblement confeccionar un de nou a partir dels existents.

-Creació detector ulls.

Per a implementar el detector d'ulls s'han de definir quins descriptors i models usar. Per això es provaran diferents estratègies i es validaran amb el *dataset*. Conseqüentment aquesta tasca depèn de la obtenció de *dataset*.

-*Profiling* dels detectors.

Per a fer un *profiling* dels detectors comprovarem la tasa d'encerts que tenen amb el data set per a obtenir un *baseline* amb el que comparar el detector final. Aquí també es podrà detectar problemes de rendiment amb els detectors, si un detector te molt poc rendiment no te sentit continuar endavant. Lògicament depèn de la creació del detector de ulls.

-Fusió detectors

Aquí implementarem i avaluarem diferents estratègies per a fusionar els detectors per a obtenir un detector de persones robust. Tot i que estrictament parlant no depèn de el *profiling*. Com hem esmentat anteriorment per evitar possibles problemes en endavant seria adient procedir amb la fusió dels detectors una vegada aquests hagin sigut avaluats.

-*Profiling* detector resultant

Amb l'objectiu de validar el detector resultant i comprovar que millora respecte els detectors individuals farem un *profiling* i el compararem amb els detectors individuals. Depèn de la fusió dels detectors.

Els recursos utilitzats per aquestes tasques son:

Recursos humans:
El director

Recursos hardware:
Ordinador personal on es portarà a terme el desenvolupament

Servidor on s'entrenaran els models

Recursos software
Microsoft Word per redactar els documents

MATLAB R2018a per desenvolupar els models

Git per organitzar el codi

7.2.3.- Fase final:

Aquesta fase consisteix en la consolidar el projecte i generar documentació.

Tasques:

-Optimització i empaquetat: Optimitzarem el detector i l'empaquetarem junt amb la seva documentació. Depèn del profiling.

-**Redacció memòria projecte:** Depèn del *profiling*.

Els recursos utilitzats per aquestes tasques son:

Recursos humans:
El director

Recursos *hardware*:
Ordinador personal on es portarà a terme el desenvolupament

Recursos software
Microsoft Word per redactar els documents

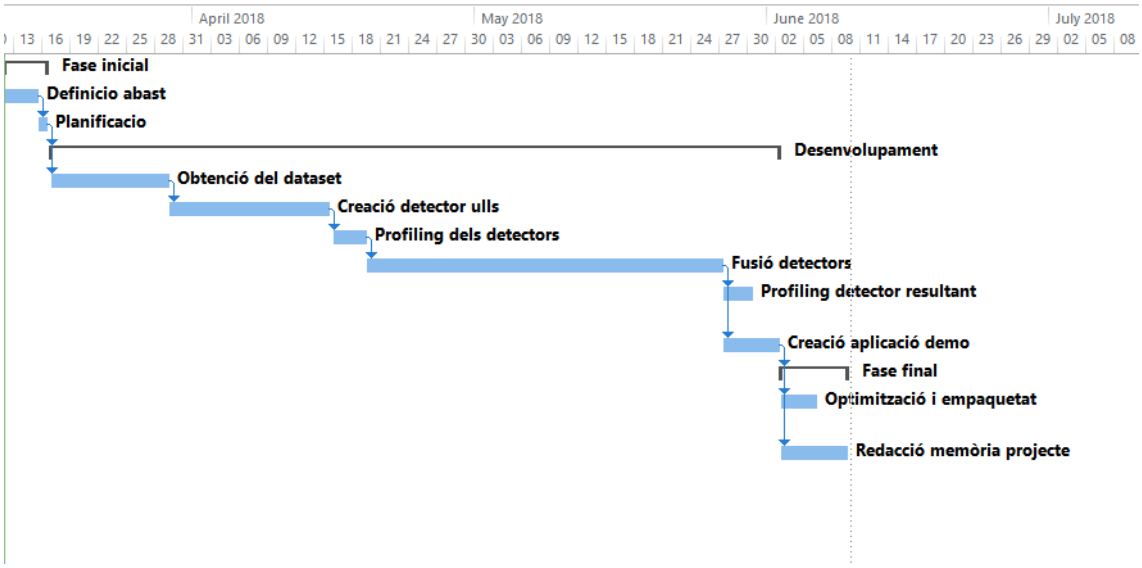
MATLAB R2018a per desenvolupar els models

Git per organitzar el codi

7.2.4 – Duració esperada de les tasques:

Tasca	Temps estimat
Definició abast	15H
Planificació	5H
Obtenció del *dataset*.	50H
Creació detector ulls.	70H
Profiling dels detectors.	15H
Fusió detectors	150H
Profiling detector resultant	15H
Optimització i empaquetat.	30H
Redacció memòria	30H
Total	405H

Per tant amb 4 hores diares el projecte es factible.



7.3.-Desviacions i alternatives:

Les metodologies àgils son particularment robustes a les possibles desviacions ja que el curts cicles de desenvolupament permeten reaccionar ràpidament a possibles problemes.

Tot i això si hi fes falta la optimització es podria sacrificar. També esmentar que tal i com esta plantejada la planificació temporal aquesta no acaba en el *deadline*. Per tant hi ha un marge petit per a desviacions en la planificació.

8.-Sostenibilitat i compromís social:

8.1-Posada en producció:

El cost ambiental a l'hora de posar en producció el projecte es limita a el impacte de la energia consumida. Ja que disposàvem del hardware no ha fet falta adquirir de nou amb el impacte ambiental i econòmic que això comporta. La duració aproximada del projecte ha sigut 3 mesos amb unes 400 de dedicació d'acord a la planificació. Amb un consum mesurat del hardware usat de 300w el consum de electricitat utilitzat per el projecte a sigut 120 KWh que correspon a 47kg de co2 emesos [10]. Degut a la naturalesa del projecte es impossible reduir el impacte sense modificar l'abast. Ja que tots els recursos utilitzats son imprescindibles per a el desenvolupament del projecte.

Econòmicament no hi ha hagut desviacions respecte al pressupost. Podria haver-se considerat altres recursos software per a reduir el cost del projecte.

8.2.-Vida útil i riscos

El projecte no te al abast la implantació del projecte. Aquest projecte es limita a desenvolupar un sistema que altres usaran amb per els seus propis fins. Per tant serà l'usuari del sistema a qui li correspongui fer el anàlisis de vida útil i riscos.

9.- Identificació de lleis i regulacions

Les lleis i regulacions rellevants per el projecte es limiten al us de les imatges de persones en el dataset. En Espanya la imatge de un mateix es considera una dada personal i per tant esta governada per la llei Orgánica 15/1999 del 13 de desembre, de protecció de dades de caràcter personal. Per tant podríem estar obligats a eliminar certes imatges del dataset sota sol·licitud del la persona en qüestió.

Ja que utilitzem datasets ja existents tenim que seguir la llicencia dels datasets. En el nostre cas nomes estem obligats a citar els dataset. Les imatges en si estan cobertes sota "Creative Commons

Attribution-NonCommercial 4.0". Aquesta llicencia permet modificar, redistribuir i usar per a qualsevol us no comercial. Si redistribuïm les imatges tenim que incloure una copia de la llicencia.

10.-Conclusió:

Donats els resultats de validació podem concloure que el projecte compleix el objectiu principal de detecció de persones robust. La estratègia de fusió fa que el sistema sigui particularment resistent a falsos positius amb una precisió del 99%. El temps d'execució del sistema es marginal per a us en un sistema de temps real i necessitaria millores per ser implantat. Possibles millores inclouen reduir la escala de detecció dels detectors o optimitzar el càlcul de característiques HOG. També es millorable el recall del sistema a un 67%.

Altres estratègies de fusió, agrupament de parts o relacionar geomètricament les parts en grup com a model constel·lació en lloc de parell en parell com es fa actualment serien camins que es podrien investigar que podrien dur a possibles millores de rendiment.

11.-Glossari

HOG – Histogram of Oriented Gradients. Un tipus de característiques per a la classificació molt usada.

LBP – Local Binary Pattern. Un tipus de característiques per a la classificació de imatges. Codifica informació de textura. Moltes vegades usada en combinació junt a HOG.

MLP – Multi Layer Perceptron. El tipus mes simple de xarxa neuronal amb capes ocultes.

ROC - Receiver Operating Characteristic. Una curva típicament usada per a representar el recall vs la tasa de errors.

Recall - fracció de les instancies detectades vs les instancies rellevants

Referencies

[1] Honda, «Asimo, The World's Most Advanced Humanoid Robot.,» [En línea]. Available: asimo.honda.com/. [Último acceso: 2 3 2018].

[2] «Kaspar the social robot,» [En línea]. Available: http://www.herts.ac.uk/kaspar . [Último acceso: 2 3 2018].

[3] R. e. a. Benenson, «"Ten Years of Pedestrian Detection, What Have We Learned?,» 2014.

[4] P. F. Felzenszwalb, R. B. Girshick y D. McAllester, «Object Detection with Discriminatively Trained Part-Based Models».

[5] «PASCAL VOC Challenge performance evaluation server.,» [En línea]. Available: http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=1 .

[6] R. e. a. Girshick, «Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.,» 2014.

[7] J. e. a. Dai, «R-FCN: Object Detection via Region-Based Fully Convolutional Networks.,» 2016.

[8] N. D. a. B. Triggs, «Histograms of Oriented Gradients for Human Detection,» 2005.

[9] «Calvin upper-Body detector,» ETH ZURICH . [En línea]. [Último acceso: 2 3 2018].

[10] [En línea]. Available: http://canviclimatic.gencat.cat/es/redueix_emissions/com-calcular-emissions-de-geh/factors_demissio_associats_a_lenergia/.

[11] «VGG Human Pose Estimation datasets,» [En línea]. Available: https://www.robots.ox.ac.uk/~vgg/data/pose/.

[12] «Buffy Stickmen V3.01 Annotated data and evaluation routines for 2D human pose estimation,» [En línea]. Available: http://www.robots.ox.ac.uk/~vgg/data/stickmen/.

[13] S. C. Z. A. Mikolajczyk K., «Human Detection Based on a Probabilistic Assembly of Robust Part Detectors,» *ECCV 2004. Lecture Notes in Computer Science, vol 3021,* 2004.