UNIVERSITAT POLITÈCNICA DE CATALUNYA

Short project

INFORME

VISIÓ PER COMPUTADOR

Autors

DAVID LATORRE
ADRIÀ AUMATELL

Índex

Introducció

La idea principal d'aquest petit projecte és la implementació d'un sistema automàtic de detecció d'ulls en una imatge, així com també diferenciar per tant aquelles imatges que contenen ulls respecte d'aquelles que no contenen ulls.

Més exactament, la idea del projecte és trobar un bon descriptor d'imatges, que funcioni per a un classificador en concret. Per tant, no es tracta de desenvolupar l'algorisme d'aprenentatge automàtic (el qual vindria a ser part més del camp de la intel·ligència artificial), sinó de la qualitat dels paràmetres diferenciadors que li passa a aquest algorisme (això és més part del camp de la visualització per computador). Hem de trobar, per tant, uns descriptors que puguin diferenciar de forma més o menys correcta aquelles imatges que són ulls d'aquelles que no en tenen, i que a la vegada vagin bé amb el classificador que volem fer servir.

El classificador que farem servir serà un classificador lineal SVM (support vector machine).

Els classificadors SVM formen part dels classificadors de predicció més robusts que hi ha avui en dia. Donat una serie d'exemples d'entrenament, cadascun marcat com a que forma part d'una certa categoria o altre, l'entrenament SVM construeix un model capaç d'assignar nous exemples a una categoria o altre, creant un classificador linear i binari no probabilistic.

L'objectiu, per tant, és trobar una sèrie de descriptors que ens pugui ajudar a diferencia entre dos clústers: Clúster No-Ulls i el cúster ulls.

Construcció de la base de dades

Per començar, cal tenir una bona base de dades d'ulls i de coses que no son ulls (no-ulls). En concret, ens fa falta 300 imatges d'ulls i 400 imatges de no-ulls. Per no complicar-nos, utilitzem les porporcionades per l'assignatura.

Per tal de poder entrenar correctament el nostre detector d'ulls cal estandartitzar la mida de les imatges i la distribució de colors. En el cas de les imatges corresponents a no-ulls cal escalar-les perquè tinguin una mida de 48x32. A més a més, cal convertir-les a escala de grisos amb 256 nivells.

Ara ja tenim les imatges de no-ulls preparades per l'entrenament, però falta processar les imatges d'ulls. Les imatges proporcionades, però, no contenen directament ulls, sinó que son imatges de cares. Així doncs, aprofitant que els ulls sempre son a la mateixa posició de les imatges, retallem l'ull dret de les 300 cares i, com hem fet amb les imatges de no-ulls, escalem i convertim a 256 nivells de gris.

Una vegada fet tot l'anterior, ja tenim la base de dades funcional i a punt per poder construir el vector de característiques.

Construcció vector de característiques

Ara que ja tenim la base de dades, ja podem començar a construir quines variables tindrà el nostre vector de característiques.

A l'hora de construir aquest vector, hem de tenir en compte de triar variables que s'adaptin correctament al tipus de classificador amb el que treballarem. En aquest cas, treballem amb un clasificador SVM, i només volem diferencia entre dos tipus de clústers. Per tant, a l'hora de triar les variables, ens interessa triar aquelles que conjuntament tinguin valors que puguin diferenciar un clúster de l'altre de forma no molt complicada.

Degut a que, al centre d'un ull sempre tenim una retina, la qual és rodona, i, en general tots els ulls sempre segueixen una forma, estaria bé considerar una variable que tingués en compte la forma de la imatge.

També, un aspecte que pot servir molt a l'hora de millorar la precisió del nostre classificador, i que s'utilitza molt en altres mètodes d'extracció de característiques (com la funció de MATLAB extractHOGFeatures), és el de subdividir la imatge en un nombre de divisions, i calcular un conjunt de característiques per a cada divisió (les característiques seran les mateixes per a cada subdivisió), de forma que finalment el vector de característiques contingui els valors de totes les característiques per a cada subdivisió. Això és especialment útil, ja que el fet de subdividir la imatge fa que ens puguem centrar més en els detalls de la imatge. Per exemple, si fem 9 subdivisions a les imatges, la subdivisió del centre, en el cas dels ulls sempre seria la retina, i per tant amb unes característiques respectivament bones això seria de gran ajuda pel classificador.

A més, si fem subdivisions, tampoc cal que el càlcul de les característiques sigui molt complex, sinó al contrari, ja que considerant que les imatges de la nostra base de dades només tenen un tamany de 48x32 pixels, les divisions tindran molts pocs pixels. Un desavantatge que cal considerar, però, a l'hora de calcular característiques per a cada subdivisió, és que el tamany total del nostre vector de

característiques pot arribar a ser bastant alt, i això, com veurem a l'apartat de la figura de mèrit, no és una cosa que ens interessi.

Per calcular el vector de característiques d'una imatge, per tant, crearem un arxiu, anomenat "mydescriptor.m", en el qual construirem una funció anomenada mydescriptor, la qual com a paràmetre rebrà una imatge de la base de dades, i retornarà el respectiu vector de característiques per a aquesta imatge.

Tenint en compte, per tant, totes les consideracions que acabem de dir sobre les característiques d'una imatge, a la funció mydescriptor hem acabat dissenyant 4 possibles creacions d'un vector de característiques. Aquests 4 mètodes diferents de construcció del vector de característiques s'assemblen en el fet que tots utilitzen les mateixes 4 principals característiques. No obstant hi ha petits canvis entre aquests, per exemple uns treballen amb la imatge original, mentre que d'altres no, i uns treballen sense divisions, mentre que d'altres no. El fet de fer aquestes 4 construccions diferents ens és útil per, més tard, a l'estudi estadístic poder comparar-les, i quedar-nos amb la millor.

Expliquem primer, per tant, quines son les 4 característiques amb les que treballen aquestes 4 construccions:

- **Mitjana de nivells de gris (de imatge o subdivisions):** Aquesta és la característica més senzilla de totes. Tot i que pugui semblar que no sigui de gran ajuda, al treballar amb imatges tan petites, i junt amb les següents característiques, veurem més tard com realment també juga un bon paper.
- **Variancia:** Es tracta d'una variable que pot anar bé junt amb la mitjana. Ja que un cop tenim la mitjana dels valors, també aniria bé saber quant acostumen a diferenciar-se el valor dels píxels d'aquesta mitjana.
- **Skew (oblicuitat):** es tracta d'una variable que mesura la falta de simetria. La skewness acostuma a ser negativa quan ens trobem amb un histograma el qual per la dreta (o per l'esquerre) és veu més torçat/desaquilibrat que per l'esquerre (o per la dreta). Pensem que aquesta variable ens pot anar bé a l'hora de diferencia components de l'ull

com la retina respecte d'imatges que no representen ulls. La formula per calcular aquesta variable és la següent:

$$\text{Skewness} = \sum_{i}^{N} (X_i - X)^3 / (N-1) * \sigma^3$$

És a dir, la divisió entre el sumatori de les diferències dels nivells de gris del histograma amb la mitjana de les intensitats de la imatge elevat a 3, i el nombre total de pixels multiplicat per la desviació estàndard de la imatge elevat a 3.

- **Kurtosis (curtosis):** Es tracta d'una mesura de si els valors dels pixels en una imatge són plans o no respecte a una distribució normal. Els conjunts de pixels, per tant, que tenen una kurtosis alta, tendeixen a tenir pics a propr de la mitjana, mentre que aquells amb una kurtosis baixa no. Es tracta d'una variable, per tant, similar a la skewness, però que ens pot ajudar a diferencia encara més els clústers.

Pel que fa a les 4 construccions diferents del vector de característiques, aquests es diferencien en:

- **1:** El primer de tots, és el més senzill, ja que el vector de característiques directament són les 4 variables acabades d'explicar, extretes de la imatge sencera original (en escala de grisos).

- **2:** Similar al primer (és a dir utilitzem la imatge original), però en aquest cas fem 9 subdivisions a la imatge. Per tant, calculem les 4 variables per a cada una de les divisions, obtenint un vector de característiques de tamany 36.
- **3:** És el mateix que el primer, però en comptes d'obtenir les característiques de la imatge original en escala de grisos, ho fem amb la imatge resultant d'aplicar-li un filtre laplacià de gauss de finestra 5x5 i sigma 0.5.
- **4:** El mateix que el segon, però amb el filtre del tercer.

La raó per la qual ens sembla que serà útil fer el filtre laplacià per als dos últims mètodes, és perquè així tenim informació sobre on hi ha els canvis de gradient a les imatges, és a dir les vores, i per tant potser podem diferenciar més el que és un ull del que no ho és.

També, a l'hora de decidir quin filtre aplicar, vam pensar en fer-ho amb un sobel, però comparant-lo amb el laplacià vam observar que donava resultats més obscurs, i per tant per l'entrenament del classificador seria pitjor. La diferència dels dos filtres la podem observar a les següents imatges:

Filtre sobel:

Filtre laplacià:

Pel que fa a esbrinar quina construcció dona més bons resultats, això ho veurem a l'apartat anàlisi estadístic, però pel fet d'utilitzar les subdivisions creiem que tant el segon com el quart mètode donaran els millors resultats d'entre els quatre.

Construcció taula de descriptors

Una vegada definit el vector de característiques cal construïr la taula de descriptors.

El procés a seguir és força simple. Primer, cal definir per cada columna de la taula, que ha de correspondre amb el nom de la característica. Seguidament, s'ha d'obtenir el vector de característiques de cada imatge de la base de dades i afegir-lo a la taula de descriptors. Finalment, per cada fila de la taula (que correspon a una imatge), cal afegir una columna el valor de la qual indiqui si la imatge descrita és, o no, ull.

Aquesta última columna serveix per poder indicar al matlab a quin tipus (ull o no-ull) pertany la imatge i així poder entrenar correctament el detector i poder generar estadístics importants (com la confusion matrix) de forma satisfactòria.

A sota es pot veure un fragment de la taula de descriptors utilitzada en la versió final del porgrama:

32 Kurtosis subimatge 8	33 Mitjana subimatge 9	34 Variancia subimatge 9	35 Skew subimatge 9	36 Kurtosis subimatge 9	37 Etiqueta
2.0842	69.8807	166.0371	0.3037	2.2062	No-Ulls
1.7558	115.7500	974.3257	-0.0145	1.7967	No-Ulls
2.6157	73.2727	688.4852	0.5161	2.9377	No-Ulls
2.3240	58.1307	2.3883e+03	1.7963	4.7167	No-Ulls
2.9671	100.9432	1.7947e+03	1.0633	3.0095	No-Ulls
2.4153	198.3295	2.3603e+03	0.0718	1.2918	No-Ulls
2.4837	136.4716	333.2792	-1.3211	4.6290	Ulls
1.9874	142.8011	228.5831	-0.0819	1.8563	Ulls
1.8400	121.8239	786.1117	0.0807	2.8475	Ulls
1.8344	125.3580	1.4608e+03	-1.1922	3.7480	Ulls
2.0896	121.9261	250.9945	0.5128	4.1258	Ulls
3.2597	109.5852	151.5241	-0.8837	3.8525	Ulls
2.5535	127.2500	853.5371	0.6059	3.4801	Ulls
10.0213	152.8864	185.4842	0.5323	4.8021	Ulls
4.0242	106.6591	197.4945	-0.0146	2.7508	Ulls
2.0610	136.6023	222.0581	-0.6754	4.3870	Ulls
3.8850	133.0170	220.1426	-0.9575	3.6874	Ulls
3.2492	128.6420	447.3854	-1.2788	3.6823	Ulls
2.9297	134.0568	807.5168	1.0005	2.9888	Ulls

Entrenament clasificador SVM

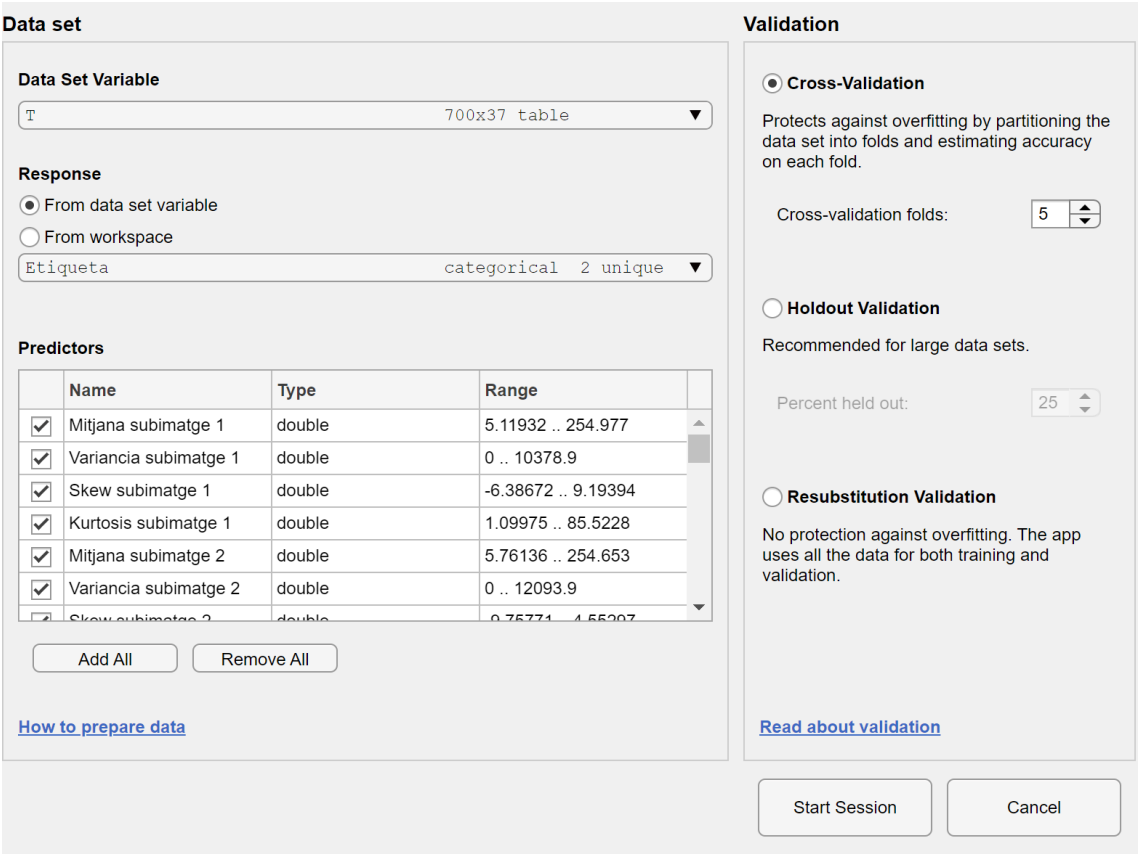
Un cop ja tenim construïda la taula de descriptors, ja tenim tot lo necessari per entrenar el nostre clasificador SVM. De fet, l'únic que necessita el classificador és la taula de descriptors, ja que aquesta ja té separada per etiquetes aquelles files que formaran part d'un clúster o d'un altre (No-Ulls, Ulls).

Farem, per tant, 4 entrenaments diferents, un per cada un dels casos diferents de la construcció del vector de característiques que hem considerat, i més endavant, compararem aquests 4 entrenaments al pròxim apartat per a veure quin és el millor.

El procediment, però, per a l'entrenament és el mateix en els 4 casos. MATLAB ens proporciona una eina bastant intuitiva, degut a que està construïda amb interfície gràfica, amb la qual podem fer l'entrenament del nostre classificador de forma ràpida, configurant aquest al nostre gust. Un cop haguem realitzat l'entrenament, en el cas que vulguem reutilitzar o ens volguem referir a aquest clasificador en el codi d'un script (per exemple quan fem les prediccions per a cada frame a l'apartat de detecció d'ulls en seqüències de videos), aquesta eina ens permet amb un simple botó, exportar la creació del classificador en una funció.

Aquesta eina s'anomena Classification Learner. Expliquem, doncs, com entrenar un classificador amb aquesta eina:

El primer que hem de fer quan l'obrim, és crear una nova sessió on li fem saber de la variable que conté el nostre vector de característiques:



En el nostre cas, tal com es pot veure a la imatge superior, la nostra taula es veu identificada pel nom T. Si ens hi fixem, podem veure a la imatge que la taula té 37 columnes, això és perquè en aquest cas entrenarem el classificador amb el nostre cas segon de construcció del vector de característiques (aquell en que dividim la imatge original en 9 subdivisions, i per a cada una de les 9 subdivisions guardem la mitjana dels valors de gris, la variància, la skew i la kurtosis), de forma que 37 = mida del vector de característiques (36) + 1 (representa l'etiqueta No-Ull, Ull).

D'aquesta taula, al classification Learner també li haurem de fer saber quina és la variable resposta, és a dir aquella que ens fa saber de la classificació dels diversos clusters. En el nostre cas, aquesta variable es diu Etiqueta, i és una variable categòrica (ja que té el valor Ull, o no-Ull).

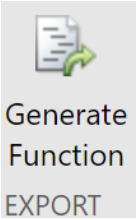
I, finalment, abans de començar la sessió, al classification Learner també li haurem de fer saber quin mètode de verificació volem, és a dir, amb quin mètode volem comprovar si el classificador s'haurà entrenat bé o no. Els dos mètodes

més populars són Cross-Validation i Holdout Validation. Holdout Validation és segurament el que s'utilitza més, sobretot per a bases de dades grans, amb el percentatge de 80% entrenament i 20% prova. No obstant, com que tenim una base de dades bastant petita, ens anirà millor una validació Cross-Validation creuada 5 cops.

Un cop fet això ja podem iniciar la sessió i realitzar un entrenament amb el classificador que vulguem. En el nostre cas utilitzarem un classificador SVM, més exactament un kernel lineal SVM, però cal destacar que el classification learner ens deixa triar entre molts altres, com: boosted trees, xarxes neuronals, boosted trees...

Un cop acabat l'entrenament, el classification learner ens proporciona una sèrie d'eines per fer-ne'n l'anàlisi, com scatter plots, confusion matrix, ROC curve... Aquestes les farem servir per al següent apartat.

Finalment, només esmentar que un cop estiguem satisfets amb un certa configuració d'entrenament, aquest entrenament el podrem passar directament com una funció escrita en codi mitjançant el següent botó:

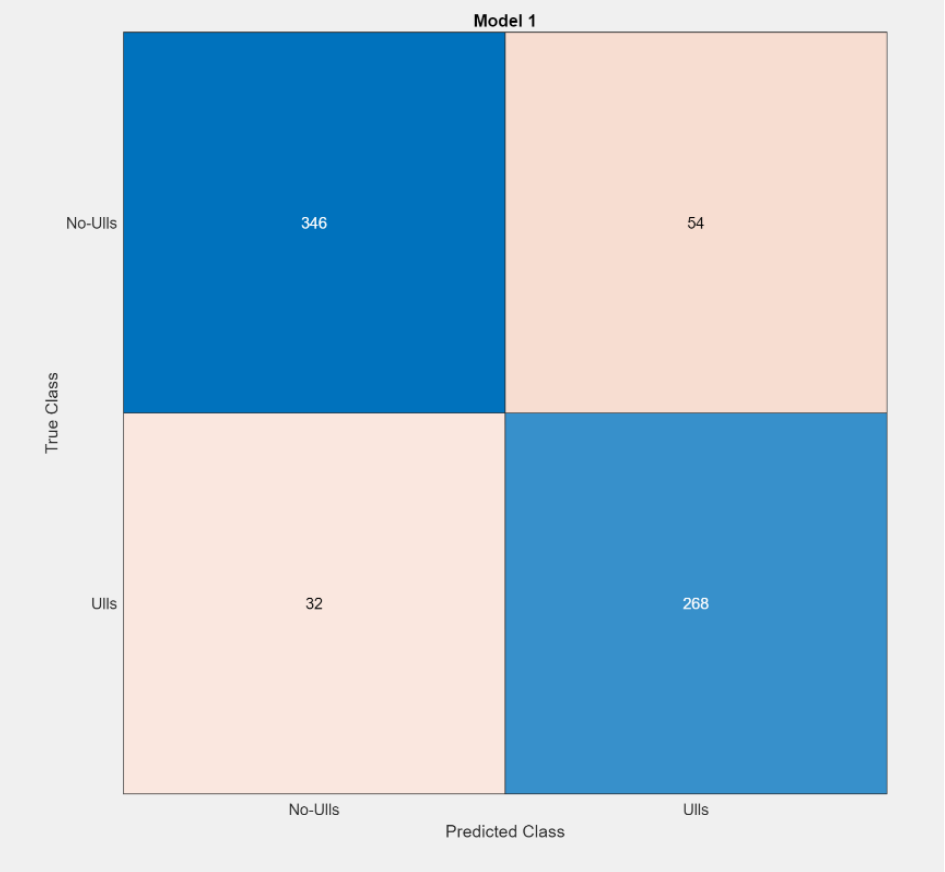


És, de fet, d'aquesta manera com creat la funció trainClassifier, amb la qual ens referim a l'arxiu project.m.

Estudi estadístic

Després d'entrenar el classificador SVM amb diverses taules de descriptors, és el moment de observar els resultats obtinguts.

Pel que fa a la primera construcció (4 característiques sobre la imatge original), obtenim la següent confusion matrix i precisió:



Training Results

Accuracy (Validation)	87.7%
Total cost (Validation)	86
Prediction speed	~12000 obs/sec
Training time	2.6809 sec

Model Type
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

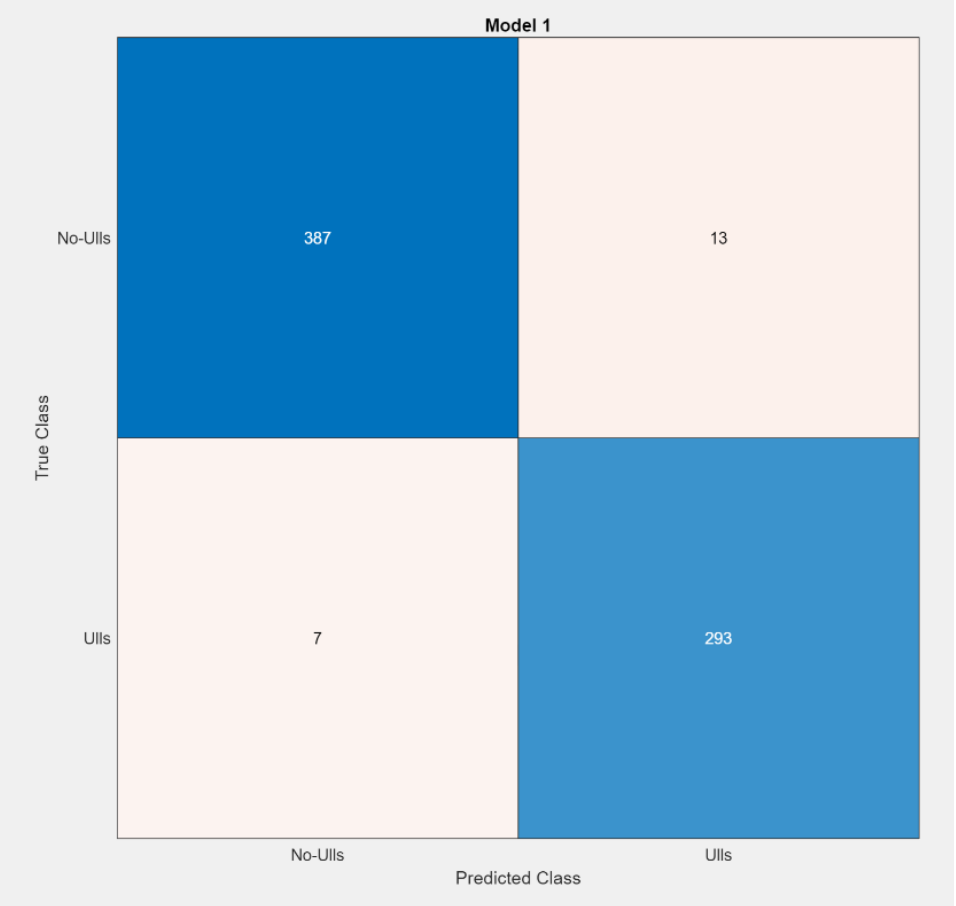
PCA
PCA disabled

Misclassification Costs
Cost matrix: default

Com es pot observar, els resultats obtinguts son força satisfactoris. L'accuracy està per sobre del 87% i la confusion matrix mostra una classificació correcta en la majoria de casos.

Malgrat tot, podem apreciar, mitjançant la confusion matrix, que aquest classificador té una taxa de falsos positius lleugerament alta (54) i que és moderadament superior a la quantitat de falsos negatius.

Anem a veure ara els resutats obtinguts amb la segona construcció (36 característiques sobre la imatge original).



Training Results

Accuracy (Validation)	97.1%
Total cost (Validation)	20
Prediction speed	~17000 obs/sec
Training time	1.4927 sec

Model Type
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

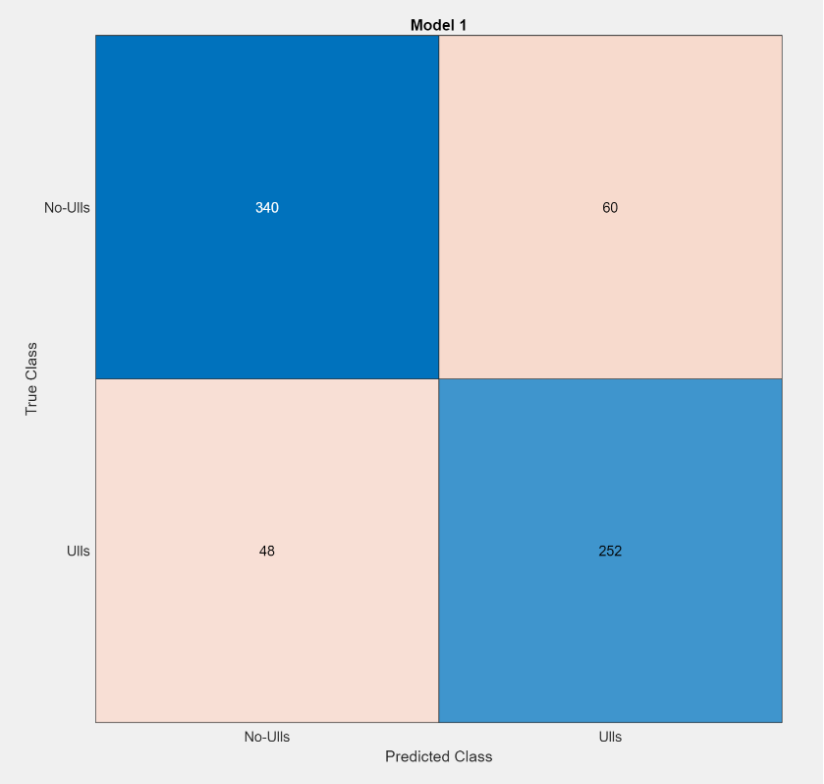
PCA
PCA disabled

Misclassification Costs
Cost matrix: default

Amb aquesta segona opció els resultats han millorat considerablement. Ara l'accuracy és del 97.1%, així que el classificador està ratllant la perfecció.

Si ens fixem en la confusion matrix, podem observar que tant el nombre de falsos positius i falsos negatius és molt petit. Tot i això, també s'observa, com el cas anterior, que hi ha més falsos positius que falsos negatius. Tot i això, el biaix no és tan important com en la primera opció.

Ara toca observar els resultats de les altres dues alternatives. Anem a observar primer els resultats obtinguts amb la tercera opció (4 descriptors sobre la imatge filtrada amb un filtre laplacià de gauss).



Model 1: Trained

Training Results

Accuracy (Validation)	84.6%
Total cost (Validation)	108
Prediction speed	~41000 obs/sec
Training time	1.1543 sec

Model Type
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

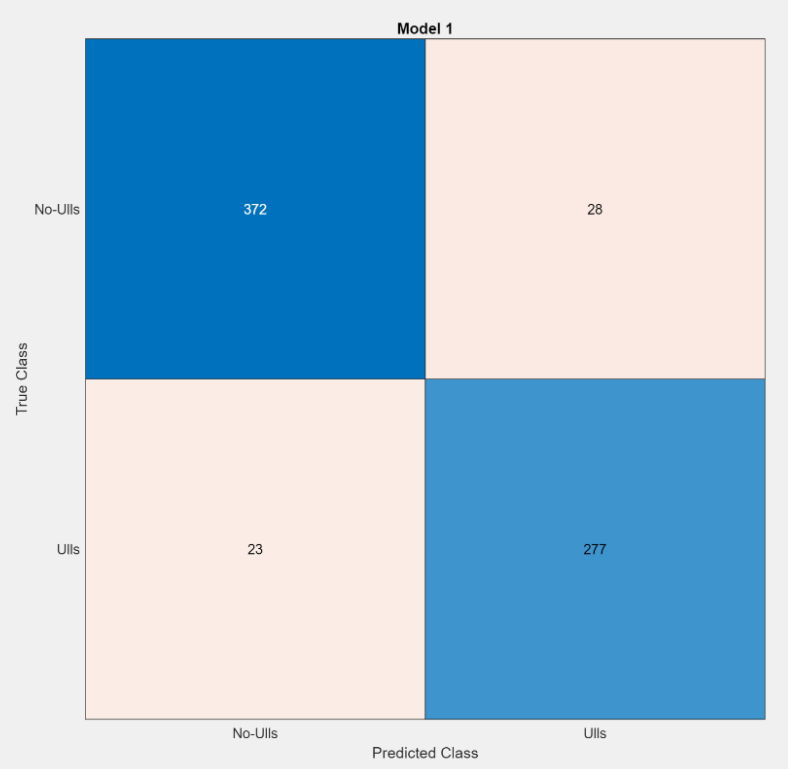
PCA
PCA disabled

Misclassification Costs
Cost matrix: default

Aquesta opció, com es pot veure, té menys accuracy que les dues anteriors (84.6%). És evident que l'estratègia de usar la imatge filtrada enlloc de l'original no resulta en majors percentatges d'exits, almenys en aquesta versió.

Si ens fixem en la confusion matrix, podem observar que segueix la tendència de les anteriors: Molts encerts i lleugerament més falsos positius que falsos negatius. També cal destacar que la utilització d'un filtre no sembla introduir cap biaix notable entre el nombre de falsos positius i negatius.

Per acabar, analitzarem la quarta opció (36 descriptors sobre la imatge filtrada amb un filtre laplacià de gauss).



Training Results

Accuracy (Validation)	92.7%
Total cost (Validation)	51
Prediction speed	~18000 obs/sec
Training time	1.1962 sec

Model Type
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: default

En aquest darrer cas, podem observar que els resultats (accuracy 92.7%) son millors que els obtinguts amb la primera i tercera opció, però son pitjors que els obtinguts amb la segona opció.

Si ens fixem amb la confusion matrix, podem veure que segueix el mateix comportament que les anteriors sense cap diferència remarcable.

En global, podem concloure que l'estratègia de partir la imatge en segments més petits funciona i millora els resultats finals. En canvi, el fet d'aplicar un filtre laplacià de gauss sobre la imatge empitjora la precisió del classificador.

A partir de les conclusions exposades, podem considerar l'opció 2 com el millor classificador pel nostre programa i, de fet, és l'implementat en el nostre codi.

Com a incís, l'accuracy obtinguda pel classificador implementat és lleugerament menor (96.69%) que l'exposada. Això és degut al fet que el model utilitzat per generar el classificador no és el mateix que l'utilitzat per mesurar-ne els resultats, tot i que ambdós models reben la mateixa taula de descriptors.

Figura de mèrit

Un cop ja hem fet l'estudi estadístic sobre els diversos casos que hem plantejat com a vectors de descriptors, ja podem calcular la figura de mèrit. És a dir, el valor que ens dirà quant de bo és realment el nostre vector de característiques quan l'utilitzem per entrenar la nostra base de dades amb l'algorisme SVM.

De l'apartat anterior, hem pogut concloure que el cas en que per calcular el vector de característiques fem 9 divisions de la imatge original i per cada una d'aquestes hi calculem 4 característiques (mitjana nivell de gris dels pixels, variancia, skewness i kurtosis) és el millor, ja que ens donava una precisió bastant millor respecte a la resta, i quasi del 100%.

No obstant, és aquest cas verdaderament el millor? Amb un vector de característiques es pot obtenir una presició molt alta, però si el tamany d'aquest vector és molt gran, això influeix negativament en el mèrit total d'aquest. No només volem obtenir un vector de característiques que ens proporcioni una presició alta, sinó que aquest contingui les mínimes variables possibles, per tal de reduir-lo al màxim i que el tamany final de la taula de característiques així com el temps d'execució d'entrenament es redueixin. És per això que, a l'hora de calcular el mèrit del nostre detector, també tenim en compte la relació de la mida del vector de característiques d'acord amb la mida de les imatges de la base de dades. La fórmula, per tant, en que hem de calcular el mèrit del nostre detector és la següent:

$$\text{Mèrit} = \sqrt{\left(1 - \frac{Mida\ del\ vector\ de\ característiques}{Mida\ de\ la\ Imatge}\right)^2 + Accuracy^2}$$

Pel que acabem d'explicar, per tant, considerem necessari també calcular el mèrit del nostre detector no només pel cas del vector de característiques amb més presició, sinó també pel primer cas de tots, en que hem obtingut una presició del 87.7% tant sols amb el vector de mida 4. Vegem quin dels dos mèrits és millor:

Mèrit segon cas (en el que la mida del vector és 36):

$$\sqrt{\left(1-\frac{36}{48\times 32}\right)^2+0.971^2}$$

Result

1.37714…

Mèrit primer cas (en el que la mida del vector és 4):

$$\sqrt{\left(1-\frac{4}{48\times 32}\right)^2+0.877^2}$$

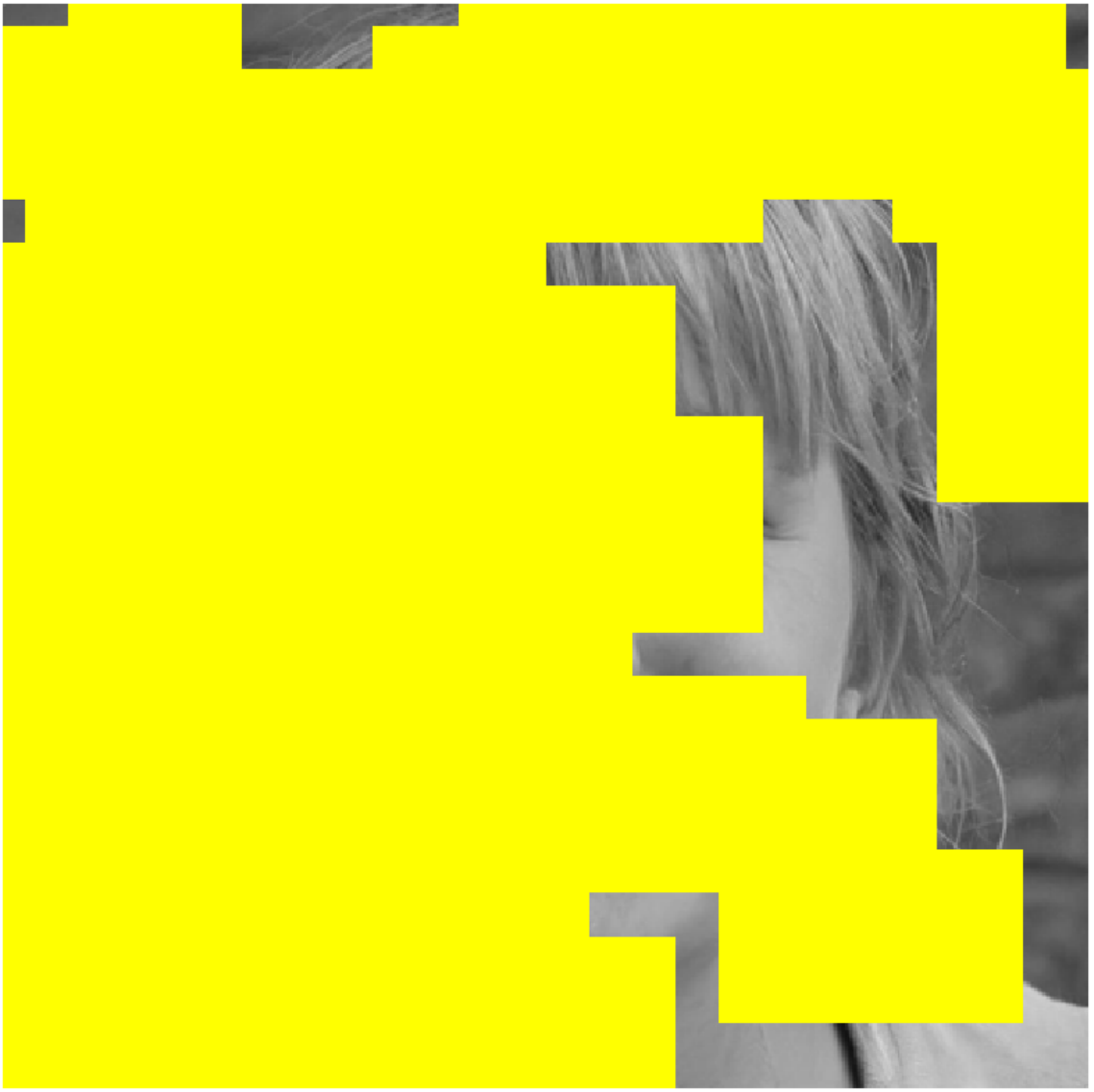
Result

1.32813…

Observem per tant, que tot i que pel segon cas la precisió és molt alta, el mèrit acaba essent molt similar degut a que disposa de moltes més variables al vector. No obstant, el segon cas és el millor, ja que té un mèrit més alt.

Detecció d'ulls en seqüències de video

Per últim, només ens falta crear el vídeo de mostra del classificador. Per detectar ulls el procés a seguir serà, per cada fotograma, detectar les cares que hi hagin i fer-hi passar una finestra lliscant per sobre. Els resultats obtinguts amb una de les cares de la base de dades son els següents:



Com es pot observar, els resultats son més que deficients. Per millorar-los de forma senzilla podem optar per modificar la tolerància del classificador. En altres paraules podem entrenar el classificador sent més estrictes amb el que considerem ull. Fent això aconseguim disminuïr el nombre de falsos positius, però també provoca un augment dels falsos negatius.

Amb el classificador menys tolerant a falsos positius obtenim el resultat següent:

És evident que tenim una millora apreciable respecte a l'anterior imatge, però és un resultat lluny de l'òptim.

Un dels possibles motius que expliquin aquests mals resultats podria ser que les imatges corresponents a no-ulls son molt generals i, si només detectem ulls a partir de la cara, es podríen afegir imatges de "no–ulls" de cabells, boques, nassos i altres parts de la casa per millorar el classificador.

El vídeo obtingut és el facilitat a l'entrega del short project.

Annex

- Fitxer project.m:

```
close all

clear

clc


%% carregar imatges cares

ImatgesCaresDir = dir('./Cares/*.jpg');

numFilesCares = length(ImatgesCaresDir);

ImatgesCares = cell(1,numFilesCares);

% mydata = zeros(numFiles);

for k = 1:numFilesCares

    ImatgesCares{k} = imread(strcat('./Cares/', ImatgesCaresDir(k).name));

    ImatgesCares{k} = rgb2gray(ImatgesCares{k});

end

for k = 1:numFilesCares

%     subplot(4,5,k);

    %figure;

    %imshow(ImatgesCares{k});

end


%% carregar imatges no ulls

%% i fer el crop en 48x32 píxels

ImatgesNoUllsDir = dir('./No_ulls/*.jpg');

numFilesNoUlls = length(ImatgesNoUllsDir);

ImatgesNoUlls = cell(1,numFilesNoUlls);

% mydata = zeros(numFiles);

for k = 1:numFilesNoUlls

    ImatgesNoUlls{k} = imread(strcat('./No_ulls/', ImatgesNoUllsDir(k).name));

    ImatgesNoUlls{k} = rgb2gray(ImatgesNoUlls{k});

    ImatgesNoUlls{k} = imresize(ImatgesNoUlls{k}, [NaN, 48]);

end

for k = 1:numFilesNoUlls

%     subplot(4,5,k);
```

```
    %figure;

    %imshow(ImatgesNoUlls{k});

end


%% Retallar imatges cares, perque nomes hi hagi ull esquerre

%% I fer el crop en 48x32 píxels

YUllEsquerre = 481;

XUllEsquerre = 385;


for k = 1:length(ImatgesCares)

    ImatgesCares{k} = ImatgesCares{k}(YUllEsquerre - 40 : YUllEsquerre + 39, ...

        XUllEsquerre - 60 : XUllEsquerre + 59, :);

    ImatgesCares{k} = imresize(ImatgesCares{k}, [NaN 48]);

end




%variable namees cas descriptors imatge original sense subdivisions

%varNames = {'Mitjana Total', 'Variancia Total', 'Skew Total', 'Kurtosis Total',
'Etiqueta'};


% variable names cas descriptors immatge original 9 subdivisions


varNames = {'Mitjana subimatge 1', 'Variancia subimatge 1', 'Skew subimatge 1',
'Kurtosis subimatge 1',...

    'Mitjana subimatge 2', 'Variancia subimatge 2', 'Skew subimatge 2', 'Kurtosis
subimatge 2',...

    'Mitjana subimatge 3', 'Variancia subimatge 3', 'Skew subimatge 3', 'Kurtosis
subimatge 3',...

    'Mitjana subimatge 4', 'Variancia subimatge 4', 'Skew subimatge 4', 'Kurtosis
subimatge 4',...

    'Mitjana subimatge 5', 'Variancia subimatge 5', 'Skew subimatge 5', 'Kurtosis
subimatge 5',...

    'Mitjana subimatge 6', 'Variancia subimatge 6', 'Skew subimatge 6', 'Kurtosis
subimatge 6',...

    'Mitjana subimatge 7', 'Variancia subimatge 7', 'Skew subimatge 7', 'Kurtosis
subimatge 7',...

    'Mitjana subimatge 8', 'Variancia subimatge 8', 'Skew subimatge 8', 'Kurtosis
subimatge 8',...
```

```
    'Mitjana subimatge 9', 'Variancia subimatge 9', 'Skew subimatge 9', 'Kurtosis
subimatge 9', 'Etiqueta'};




% variable names cas imatge laplacia-gauss sense subdivisions

%varNames = {'Mitjana Total laplacia', 'Variancia Total laplacia', 'Skew Total
laplacia', 'Kurtosis Total laplacia', 'Etiqueta'};



% variable names cas imatge laplacia-gauss 9 subdivisions

%{

varNames = {'Mitjana laplacia subimatge 1', 'Variancia laplacia subimatge 1', 'Skew
laplacia subimatge 1', 'Kurtosis laplacia subimatge 1',...

    'Mitjana laplacia subimatge 2', 'Variancia laplacia subimatge 2', 'Skew laplacia
subimatge 2', 'Kurtosis laplacia subimatge 2',...

    'Mitjana laplacia subimatge 3', 'Variancia laplacia subimatge 3', 'Skew laplacia
subimatge 3', 'Kurtosis laplacia subimatge 3',...

    'Mitjana laplacia subimatge 4', 'Variancia laplacia subimatge 4', 'Skew laplacia
subimatge 4', 'Kurtosis laplacia subimatge 4',...

    'Mitjana laplacia subimatge 5', 'Variancia laplacia subimatge 5', 'Skew laplacia
subimatge 5', 'Kurtosis laplacia subimatge 5',...

    'Mitjana laplacia subimatge 6', 'Variancia laplacia subimatge 6', 'Skew laplacia
subimatge 6', 'Kurtosis laplacia subimatge 6',...

    'Mitjana laplacia subimatge 7', 'Variancia laplacia subimatge 7', 'Skew laplacia
subimatge 7', 'Kurtosis laplacia subimatge 7',...

    'Mitjana laplacia subimatge 8', 'Variancia laplacia subimatge 8', 'Skew laplacia
subimatge 8', 'Kurtosis laplacia subimatge 8',...

    'Mitjana laplacia subimatge 9', 'Variancia laplacia subimatge 9', 'Skew laplacia
subimatge 9', 'Kurtosis laplacia subimatge 9', 'Etiqueta'};

%}



%% Calcular taula descriptors

T = array2table(mydescriptor(ImatgesNoUlls{1}));

T.Etiqueta = categorical({'No-Ulls'});

T.Properties.VariableNames = varNames;



for k = 2:length(ImatgesNoUlls)

    temp = array2table(mydescriptor(ImatgesNoUlls{k}));

    temp.Etiqueta = categorical({'No-Ulls'});

    temp.Properties.VariableNames = varNames;

    T = [T;temp];

end
```

```
for k = 1:length(ImatgesCares)
    temp = array2table(mydescriptor(ImatgesCares{k}));
    temp.Etiqueta = categorical({'Ulls'});
    temp.Properties.VariableNames = varNames;
    T = [T;temp];
end



%% Training
[trainedClassifier, validationAccuracy] = trainClassifier(T);
validationAccuracy



trainClassifier(T)
```

- Fitxer mydescriptor.m

```
%% input imatge escala grisos 48x32
function [D] = mydescriptor(I)
%A = fspecial('sobel');
%figure
%imshow(imfilter(I, A));
%B = fspecial('log', [5 5], 0.35);
%figure
%imshow(imfilter(I, B));
%% possibles features
    % laplatian of gaussian per a 6 subdivisions
        % de cada una,
        % mitjana nivells gris
        % tambe variancia
        % tambe skewImatgeness
        % tambe kurtosisImatge
```

```
   % fer aixo tambe per a la imatge original?




% prova mitjana nivells gris, variancia, skew i kurtsosis

% sense subdivisions imatge origianl

    %{

    D = [];

    % histograma

    [valorsHist, nivellsGris] = imhist(I);

    nombrePixels = sum(valorsHist);

    % Mitjana nivells de gris

    mitjanaImatge = sum(nivellsGris .* valorsHist) / nombrePixels;

    D(end+1) = mitjanaImatge;

    % Variancia imatge

    varianciaImatge = sum((nivellsGris - mitjanaImatge) .^ 2 .* valorsHist) /
(nombrePixels-1);

    D(end+1) = varianciaImatge;

    % Desviacio estandard imatge

    desEstandardImatge = sqrt(varianciaImatge);

    % skewImatge image.

    skewImatge = sum((nivellsGris - mitjanaImatge) .^ 3 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^3);

    D(end+1) = skewImatge;

    % kurtosisImatge image.

    kurtosisImatge = sum((nivellsGris - mitjanaImatge) .^ 4 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^4);

    D(end+1) = kurtosisImatge;

    %}




%prova prova mitjana nivells gris, variancia, skew i kurtsosis

% amb 6 subdivisions imatge original


   D = [];

   for i = 1:3

       for j = 1:3
```

```
           subimage = I(round(((i-1)*(32/3))+1):round(i*(32/3)),
round((j-1)*(48/3)+1):round(j*(48/3)));

            % histograma

            [valorsHist, nivellsGris] = imhist(subimage);

            nombrePixels = sum(valorsHist);

            % Mitjana nivells de gris

            mitjanaImatge = sum(nivellsGris .* valorsHist) / nombrePixels;

            D(end+1) = mitjanaImatge;

            % Variancia imatge

            varianciaImatge = sum((nivellsGris - mitjanaImatge) .^ 2 .* valorsHist) /
(nombrePixels-1);

            D(end+1) = varianciaImatge;

            % Desviacio estandard imatge

            desEstandardImatge = sqrt(varianciaImatge);

            % skewImatge image.

            skewImatge = sum((nivellsGris - mitjanaImatge) .^ 3 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^3);

            D(end+1) = skewImatge;

            % kurtosisImatge image.

            kurtosisImatge = sum((nivellsGris - mitjanaImatge) .^ 4 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^4);

            D(end+1) = kurtosisImatge;

       end

   end







%prova prova mitjana nivells gris, variancia, skew i kurtsosis

% imatge laplatian of gaussian sense subdivisions

    %{

    filtreLaplatianOfGaussian = fspecial('log', [5 5], 0.50);

    ImageLaplatian = imfilter(I, filtreLaplatianOfGaussian);

    %figure

    %imshow(ImageLaplatian);



    D = [];

    % histograma
```

```
    [valorsHist, nivellsGris] = imhist(ImageLaplatian);

    nombrePixels = sum(valorsHist);

    % Mitjana nivells de gris

    mitjanaImatge = sum(nivellsGris .* valorsHist) / nombrePixels;

    D(end+1) = mitjanaImatge;

    % Variancia imatge

    varianciaImatge = sum((nivellsGris - mitjanaImatge) .^ 2 .* valorsHist) /
(nombrePixels-1);

    D(end+1) = varianciaImatge;

    % Desviacio estandard imatge

    desEstandardImatge = sqrt(varianciaImatge);

    % skewImatge image.

    skewImatge = sum((nivellsGris - mitjanaImatge) .^ 3 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^3);

    D(end+1) = skewImatge;

    % kurtosisImatge image.

    kurtosisImatge = sum((nivellsGris - mitjanaImatge) .^ 4 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^4);

    D(end+1) = kurtosisImatge;

    %}



%prova prova mitjana nivells gris, variancia, skew i kurtsosis

% imatge laplatian of gaussian 6 subdivisions

    %{

    filtreLaplatianOfGaussian = fspecial('log', [5 5], 0.50);

    ImageLaplatian = imfilter(I, filtreLaplatianOfGaussian);

    %figure

    %imshow(ImageLaplatian);



   D = [];

   for i = 1:3

       for j = 1:3

           subimage = ImageLaplatian(((i-1)*(32/3))+1:(i*(32/3)),
((j-1)*(48/3))+1:(j*(48/3)));

            % histograma

            [valorsHist, nivellsGris] = imhist(subimage);

            nombrePixels = sum(valorsHist);
```

```
            % Mitjana nivells de gris

            mitjanaImatge = sum(nivellsGris .* valorsHist) / nombrePixels;

            D(end+1) = mitjanaImatge;

            % Variancia imatge

            varianciaImatge = sum((nivellsGris - mitjanaImatge) .^ 2 .* valorsHist) /
(nombrePixels-1);

            D(end+1) = varianciaImatge;

            % Desviacio estandard imatge

            desEstandardImatge = sqrt(varianciaImatge);

            % skewImatge image.

            skewImatge = sum((nivellsGris - mitjanaImatge) .^ 3 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^3);

            D(end+1) = skewImatge;

            % kurtosisImatge image.

            kurtosisImatge = sum((nivellsGris - mitjanaImatge) .^ 4 .* valorsHist) /
((nombrePixels - 1) * desEstandardImatge^4);

            D(end+1) = kurtosisImatge;

       end

   end

    %}


end
```

- Fitxer trainClassifier.m (generat automàticament per MATLAB):

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)

% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)

% Returns a trained classifier and its accuracy. This code recreates the

% classification model trained in Classification Learner app. Use the

% generated code to automate training the same model with new data, or to

% learn how to programmatically train models.

%
```

```
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy as a
%       percentage. In the app, the Models pane displays this overall
%       accuracy score for each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 13-Jun-2022 14:57:35
```

```
% Extract predictors and response

% This code processes the data into the right shape for training the

% model.

inputTable = trainingData;

predictorNames = {'Mitjana subimatge 1', 'Variancia subimatge 1', 'Skew subimatge 1',
'Kurtosis subimatge 1', 'Mitjana subimatge 2', 'Variancia subimatge 2', 'Skew
subimatge 2', 'Kurtosis subimatge 2', 'Mitjana subimatge 3', 'Variancia subimatge 3',
'Skew subimatge 3', 'Kurtosis subimatge 3', 'Mitjana subimatge 4', 'Variancia
subimatge 4', 'Skew subimatge 4', 'Kurtosis subimatge 4', 'Mitjana subimatge 5',
'Variancia subimatge 5', 'Skew subimatge 5', 'Kurtosis subimatge 5', 'Mitjana
subimatge 6', 'Variancia subimatge 6', 'Skew subimatge 6', 'Kurtosis subimatge 6',
'Mitjana subimatge 7', 'Variancia subimatge 7', 'Skew subimatge 7', 'Kurtosis
subimatge 7', 'Mitjana subimatge 8', 'Variancia subimatge 8', 'Skew subimatge 8',
'Kurtosis subimatge 8', 'Mitjana subimatge 9', 'Variancia subimatge 9', 'Skew
subimatge 9', 'Kurtosis subimatge 9'};

predictors = inputTable(:, predictorNames);

response = inputTable.Etiqueta;

isCategoricalPredictor = [false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false];



% Train a classifier

% This code specifies all the classifier options and trains the classifier.

classificationSVM = fitcsvm(...

    predictors, ...

    response, ...

    'KernelFunction', 'linear', ...

    'PolynomialOrder', [], ...

    'KernelScale', 'auto', ...

    'BoxConstraint', 1, ...

    'Standardize', true, ...

    'ClassNames', categorical({'No-Ulls'; 'Ulls'}));



% Create the result struct with predict function

predictorExtractionFcn = @(t) t(:, predictorNames);

svmPredictFcn = @(x) predict(classificationSVM, x);

trainedClassifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));



% Add additional fields to the result struct

trainedClassifier.RequiredVariables = {'Kurtosis subimatge 1', 'Kurtosis subimatge
2', 'Kurtosis subimatge 3', 'Kurtosis subimatge 4', 'Kurtosis subimatge 5', 'Kurtosis
subimatge 6', 'Kurtosis subimatge 7', 'Kurtosis subimatge 8', 'Kurtosis subimatge 9',
'Mitjana subimatge 1', 'Mitjana subimatge 2', 'Mitjana subimatge 3', 'Mitjana
```

```
subimatge 4', 'Mitjana subimatge 5', 'Mitjana subimatge 6', 'Mitjana subimatge 7',
'Mitjana subimatge 8', 'Mitjana subimatge 9', 'Skew subimatge 1', 'Skew subimatge 2',
'Skew subimatge 3', 'Skew subimatge 4', 'Skew subimatge 5', 'Skew subimatge 6', 'Skew
subimatge 7', 'Skew subimatge 8', 'Skew subimatge 9', 'Variancia subimatge 1',
'Variancia subimatge 2', 'Variancia subimatge 3', 'Variancia subimatge 4', 'Variancia
subimatge 5', 'Variancia subimatge 6', 'Variancia subimatge 7', 'Variancia subimatge
8', 'Variancia subimatge 9'};

trainedClassifier.ClassificationSVM = classificationSVM;

trainedClassifier.About = 'This struct is a trained model exported from
Classification Learner R2021b.';

trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use:
\n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name of the variable that is
this struct, e.g. ''trainedModel''. \n \nThe table, T, must contain the variables
returned by: \n  c.RequiredVariables \nVariable formats (e.g. matrix/vector,
datatype) must match the original training data. \nAdditional variables are ignored.
\n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'',
''stats.map''), ''appclassification_exportmodeltoworkspace'')">How to predict using
an exported model</a>.');



% Extract predictors and response

% This code processes the data into the right shape for training the

% model.

inputTable = trainingData;

predictorNames = {'Mitjana subimatge 1', 'Variancia subimatge 1', 'Skew subimatge 1',
'Kurtosis subimatge 1', 'Mitjana subimatge 2', 'Variancia subimatge 2', 'Skew
subimatge 2', 'Kurtosis subimatge 2', 'Mitjana subimatge 3', 'Variancia subimatge 3',
'Skew subimatge 3', 'Kurtosis subimatge 3', 'Mitjana subimatge 4', 'Variancia
subimatge 4', 'Skew subimatge 4', 'Kurtosis subimatge 4', 'Mitjana subimatge 5',
'Variancia subimatge 5', 'Skew subimatge 5', 'Kurtosis subimatge 5', 'Mitjana
subimatge 6', 'Variancia subimatge 6', 'Skew subimatge 6', 'Kurtosis subimatge 6',
'Mitjana subimatge 7', 'Variancia subimatge 7', 'Skew subimatge 7', 'Kurtosis
subimatge 7', 'Mitjana subimatge 8', 'Variancia subimatge 8', 'Skew subimatge 8',
'Kurtosis subimatge 8', 'Mitjana subimatge 9', 'Variancia subimatge 9', 'Skew
subimatge 9', 'Kurtosis subimatge 9'};

predictors = inputTable(:, predictorNames);

response = inputTable.Etiqueta;

isCategoricalPredictor = [false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false];



% Perform cross-validation

partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold', 5);



% Compute validation predictions

[validationPredictions, validationScores] = kfoldPredict(partitionedModel);



% Compute validation accuracy

validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

- Fitxer decideixSiUll.m

```
function [b] = decideixSiUll (I,nomVariables,classificador)
    %res = imresize(I,[32 48]);
    %imshow(I);
    DI = mydescriptor(I);
    Taula = array2table(DI);
        Taula.Properties.VariableNames = nomVariables;
        pred = classificador.predictFcn(Taula);
        if pred == "No-Ulls"
            b = 0;
        else
            b = 1;
        end
end
```

- Fitxer detectaUlls.m

```
function [imatgeOut] = detectaUlls(imatgeIn, classificador,nomsVar)

imatgeInRes = imresize(imatgeIn,[NaN, 250]);
N= 48;
M = round(N*2/3);

%%configurem la mida dels blocs perquè la mida sigui M x N (48x32)

step = 15;

h = round((M-step)/2); %round(r(4)/2 - step/2);
w = round((N-step)/2); %round(r(3)/2 - step/2);

varNames2 = nomsVar(1:length(nomsVar)-1);

blk = [h,w];
```

```
R = blkproc(imatgeInRes, [step step], blk, @decideixSiUll, varNames2, classificador);

R = imresize(R,size(imatgeIn));

R = R > 0.1;

cc = bwconncomp(R);

imatgeOut = regionprops(cc,'BoundingBox');


% B = imoverlay(imatgeIn,R,'y');

% figure

% imshow(B);




end
```

- Fitxer video.m:

```
%% crea el detector si no existeix

if exist('trainedClassifier','var') == 0

    project

end



%% Computar video


% Create a cascade detector object.

Detector = vision.CascadeObjectDetector('FrontalFaceLBP');


% Read a video frame and run the face detector.

videoReader = VideoReader('video.webm');


videoOut = VideoWriter('video-out.avi'); %create the video object


open(videoOut); %open the file for writing


a=0;
```

```
while hasFrame(videoReader)

    % get the next frame

    videoFrame = readFrame(videoReader);

    bbox = step(Detector, videoFrame);


    for k = 1:size(bbox)

        subimage = imcrop(videoFrame, bbox(k,:));

        ulls = detectaUlls(rgb2gray(subimage),trainedModel4,varNames);

        quadrat=zeros(size(ulls,1),4);

        for u=1:size(ulls,1)

            quadrat(u,1) = round(ulls(u).BoundingBox(1) +bbox(k,1));

            quadrat(u,2) = round(ulls(u).BoundingBox(2) +bbox(k,2));

            quadrat(u,3) = ulls(u).BoundingBox(3);

            quadrat(u,4) = ulls(u).BoundingBox(4);

        end

            %quadrats

            videoFrame = insertShape(videoFrame, 'Rectangle', quadrat);

    end


    writeVideo(videoOut,videoFrame); %write the image to file


%     figure

%     imshow(videoFrame)

    a=a+1;

    a


end


close(videoOut); %close the file
```