UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Visió Per Computador

Reconeixement automàtic de senyals de trànsit

Marc Monfort Grau

David Santos Plana

11 de gener de 2021

Índex

1 Introducció

L'objectiu del projecte és extreure i analitzar diferents descriptors d'imatge per poder classificar els 43 senyals de trànsit utilitzant algorismes d'aprenentatge automàtic.

2 Descriptors

A continuació detallarem els diferents descriptors utilitzats per modelar les característiques dels senyals de trànsit. Hem utilitzat descriptors locals, ja que considerem que pel nostre conjunt d'imatges, on ja tenim els senyals localitzats, els descriptors locals són els més adequats.

2.1 Percentatge de Color

El color és una característica molt distintiva entre senyals. El nostre conjunt de senyals es poden dividir en 5 subgrups depenent del seu color.

- Vermelles
    - Contorn
    - Superfície
- Blaves
- Grogues
- Negres (Ausencia dels altres colors)

Per descriure aquesta característica, calcularem per cada imatge el percentatge de píxels, amb els colors corresponent als colors distintius de cada grup de senyals, respecte al total de píxels de la imatge. Utilitzant el percentatge (en lloc d'un histograma) ens permet descriure imatges de diferents mides, de forma equivalent (tenint en compte que les imatges del nostre *dataset* estan retallades amb una proporció similar de senyal respecte a la quantitat de fons).

Pel càlcul del percentatge de color, primer convertim la imatge a HSV, i obtenim el nombre de píxels que tenen el valor de HUE i de Saturació corresponent a l'interval dels quatre colors principals. La conversió a HSV permet ser invariant a canvis d'il·luminació.

El fitxer *getColorFeature.m* conte la implementació.

2.2 Transformada de Hough

Junt amb el color, la forma és l'altra característica més distintiva entre els diferents grups de senyals. Entre el nostre conjunt d'imatges podem distingir 5 formes diferents de senyals.

- Circular
- Triangular
- Triangular (invertida)
- Rombe
- Octagonal

Un mètode per descriure la forma d'un senyal és utilitzant la transformada de Hough. Aquesta tècnica ens permet trobar figures expressades matemàticament en una imatge.

La funció de MATLAB *imfindcircles*, realitza la transformada de Hough per obtenir la posició i la mètrica dels cercles trobats a la imatge, especificant com a paràmetre el radi del cercle que volem trobar. Utilitzant aquesta funció podem distingir entre imatges amb senyals circulars i la resta d'imatges amb senyals no circulars.

Per implementar aquest descriptor cridem a la funció *imfindcircles*, especificant com radi del cercle, un interval entre la mida de la imatge dividit per 5 i la mida de la imatge dividit per 2 (així només considerem un radi adequat a la mida de la imatge que estem analitzant). El descriptor obtingut serà el valor de la mètrica (que semblant és a un cercle perfecte) del cercle més perfecte trobat i que tingui el seu centre proper al centre de la imatge.

2.3 Histograma de gradients orientats (HOG)

Una altra manera de descriure la forma dels senyals és utilitzant els histogrames de gradients orientats (HOG). Aquest mètode consisteix a calcular el gradient i l'orientació de les vores per diferents subdivisions de la imatge, i generar un histograma per cadascuna d'aquestes regions.

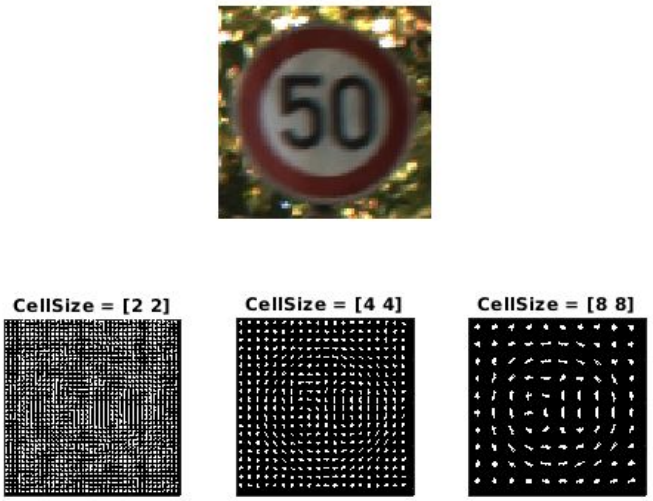


Figura 1. HOG

En la Figura 1, podem veure el resultat d'aplicar HOG a un senyal amb diferents mides de subdivisions. Depenent d'aquestes mides som capaços de distingir, amb més o menys detall, la forma i el contingut interior de la senyal.

La implementació per obtenir els descriptors de HOG ha sigut utilitzant la funció de MATLAB extractHOGFeatures, que ens retorna el vector de característiques. Com a paràmetre especifiquem la mida adequada de les subdivisions.

Ens hem trobat el problema que HOG no és invariant a l'esclat de la imatge, ja que per diferents resolucions obtenim una mida diferent del vector de característiques. La solució passa a modificar la mida de totes les imatges a un valor fix. Amb aquesta solució obtenim un vector de característiques amb el mateix nombre d'elements, però disminueix la resolució d'algunes imatges, i la qualitat del descriptor.

2.4 Speeded Up Robust Feature (SURF)

De manera similar, els descriptors de SURF permeten descriure una imatge a partir dels seus punts d'interès (contorns, vèrtex). El benefici d'aplicar aquesta tècnica en comparació a HOG, és que el SURF sí que és invariant a l'escalat. Això ens anirà molt bé en el nostre cas, ja que les imatges dels senyals són d'escales molt diverses.

L'algorisme consisteix a seleccionar un conjunt de punts d'interès obtinguts a partir dels gradients de la imatge, i genera una descripció de cada punt considerant la seva àrea circumdant. El descriptor final de la imatge és la unió dels descriptors dels punts d'interès trobats.

Figura 2. SURF

En la Figura 2, podem veure els 4 punts més significatius que ha trobat SURF per poder descriure la imatge amb el senyal de velocitat 50.

El problema d'aquest algorisme és que el nombre de punts d'interès pot variar depenent de cada imatge, i aleshores tindrem un nombre diferent de descriptors per diferents imatges. I un altre problema és que tot i tenir el mateix nombre de punts d'interès i descriptors, faria falta una forma d'entrenar un classificador per poder comparar la semblança entre diversos punts d'interès de diferents imatges.

La solució és fer servir la tècnica de 'bag of visual words'. Aquesta tècnica consisteix a generar un histograma del nombre d'ocurrències dels descriptors de cada punt. Per poder reduir la mida d'aquest histograma, s'aplica l'algorisme de K-means per obtenir un nombre K de classes. Per cada punt d'interès, s'augmenta el nombre d'elements de la classe més propera utilitzant K-means.

Per implementar la tècnica de 'bag of visual words', hem fet servir la funció de MATLAB *bagOfFeatures*, que per defecte ja utilitza SURF. Aquesta funció ens retorna un objecte *bag* que conté la informació del vocabulari utilitzat per descriure les imatges (els centroides de les K subclasses del K-means dels descriptors). Amb aquest objecte *bag*, podem fer servir la funció *encode* per obtenir el vector de característiques de qualsevol imatge utilitzant el vocabulari generat prèviament.

3 Classificadors

Existeixen molts tipus d'algorismes d'aprenentatge automàtic que es poden utilitzar per classificar. En el curs de Visió per Computador només hem pogut estudiar un conjunt molt petit d'aquests algorismes. Aquests seran els que analitzarem i amb els que experimentarem amb més detall.

Cal tenir en compte que en l'actualitat els algorismes més utilitzats pel reconeixement d'imatges fan servir xarxes neuronals. Utilitzant algorismes d'aprenentatge profund s'aconsegueixen molts bons resultats en el reconeixement, i faciliten la feina en l'extracció de característiques de la imatge, ja que és possible entrenar una xarxa neuronal utilitzant directament els píxels de les imatges, sense haver de pensar en com extreure característiques rellevants. Per al nostre estudi, no analitzarem cap algorisme d'aprenentatge profund, ja que no és l'objectiu d'aquest treball, tot i que s'ha de tenir present que aquests algorismes podrien donar millors resultats.

A continuació descrivim molt breument un conjunt dels algorismes de classificació utilitzats en l'experimentació realitzada en el següent apartat.

3.1 K-Nearest Neighbours

Aquest algorisme consisteix en mapejar totes les mostres segons el valor de cada mostra, de tal manera que cada imatge nova que li passem la mapejara també, i buscarà els K veïns més propers, i d'aquests decidirà a quina classe pertany segons la majoria.

3.2 Decision Tree

Aquest algorisme es basa a separar els elements d'una classe amb els de la resta mitjançant restriccions com per exemple: si ***valorDeX > threshold*** llavors **X** pertany a un grup, si no a un altre. Després continua afegint condicions per tal de separar en més subgrups. Acabem formant un arbre en què a cada vector de característiques de cada imatge li aplicarà les condicions pertinents, i quan arribi a una fulla, aquí es decidirà a quina classe pertany, la qual serà la classe majoritària del subgrup pertanyent a aquella fulla.

3.3 Support Vector Machine

Aquest algorisme és bastant complex. Resumint una mica, el que fa és crear un hiperplà o un conjunt d'hiperplans (depenent del nombre de classes) dintre d'un espai vectorial de la dimensionalitat del nombre de variables. Això permet crear espais que distingeixen entre classes, on cada element a predir serà ubicat en l'espai vectorial, i identificat segons la zona on estigui.

4 Experiments

En aquest apartat analitzarem per cada descriptor el rendiment que tenen per segmentar els diferents senyals de trànsit. La forma de trobar el millor classificador és provar-los tots, i quedar-nos el que ens doni millors resultats. Farem ús de l'App de MATLAB *Classification Learner*. Aquesta eina ens facilita entrenar els diferents models de classificació i compara el seu rendiment.

Utilitzarem el codi getClassification.m per obtenir la matriu de característiques del nostre *dataset*. Cada fila representa una imatge del nostre dataset, i cada columna descriu una característica de la imatge. Totes les imatges tenen el mateix nombre de característiques.

No utilitzarem totes les imatges del *dataset*, ja que les imatges han sigut obtingudes dels fotogrames capturats per un vehicle, i les imatges de fotogrames successius són gairebé idèntiques i no ens beneficiaria fer servir totes les imatges, ja que probablement obtindríem *overfitting*. Per aquest motiu, només considerarem 3 imatges separades dels fotogrames capturats (0, 15, 29).

Per entrenar el classificador utilitzarem la tècnica de *cross-validation* amb 10 *folds*, ja que en general dona millors resultats que una partició de *TrainingSet i TestSet*, tot i que requereix més temps de computació.

4.1 Percentatge de Color

El descriptor del percentatge de color només ens permet dividir els senyals en els 5 subgrups ja comentats. Per aquest motiu, en l'entrenament del classificador, identificarem cada imatge amb el subgrup al qual pertany. D'aquest forma, podrem avaluar si realment el descriptor ens permet classificar les imatges pel seu color.

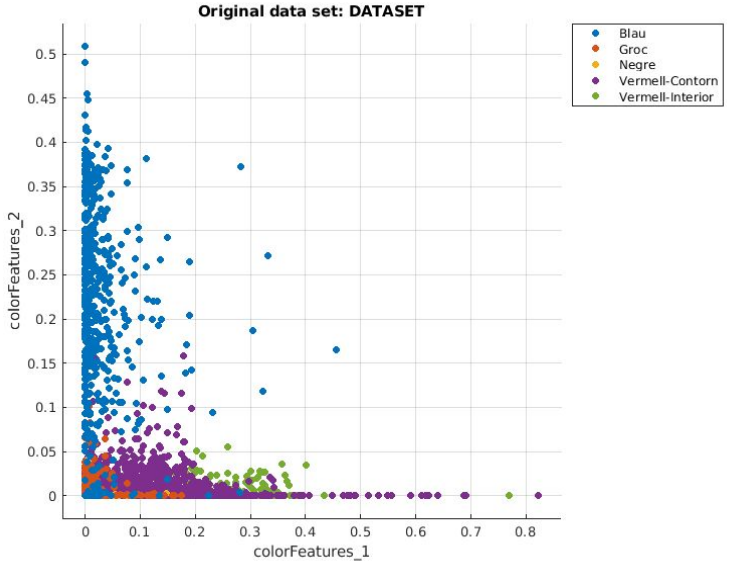


Figura 3. Scatter Plot

A la Figura 3 s'observa la distribució de les imatges depenent de la seva classe en una gràfica del percentatge de vermell i blau. Tot i que només estem veient 2 dels 4 valors del descriptor de color, ja som capaços d'apreciar una divisió en regions dels subgrups de senyals.

Després de realitzar vàries proves amb diferents algorismes de classificació, obtenim que el millor resultat ha sigut amb un SVM Quadratic, amb una precisió del 87.3%.

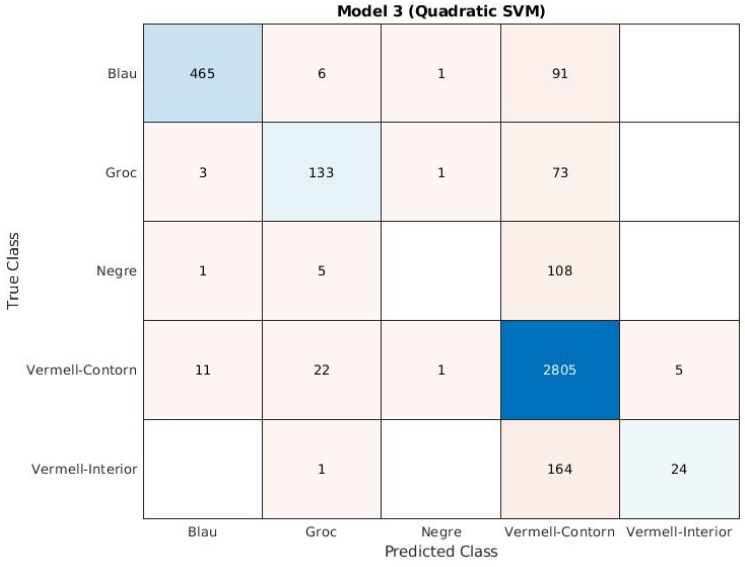


Figura 4. Confusion Matrix

La Figura 4 mostra la matriu de confusió obtinguda amb l'algorisme SVM. Veiem que tot i que els resultats per les classes "Vermell-Contorn", "Blau" i "Groc" són bastant positius, per a la resta de classes els resultats no són gaire bons. Hem de tenir en compte que si agrupem la classe "Vermell-Contorn" amb "Vermell-Interior", els resultats serien més positius, però el nostre objectiu és trobar el nombre de divisions més gran que permet el descriptor. També veiem que no aconseguim classificar correctament la classe "Negre", i haurem de trobar un millor descriptor per complementar al percentatge de color.

Concloem que el descriptor del percentatge de color és força positiu, i serà inclòs en el classificador final.

4.2 Transformada de Hough Circular

El vector de característiques obtingut amb el descriptor de la transformada de Hough circular, és un únic valor que representa la mètrica del cercle més perfecte trobat a la imatge. Amb aquest únic valor hauríem de poder segmentar entre imatges circulars i imatges no circulars.

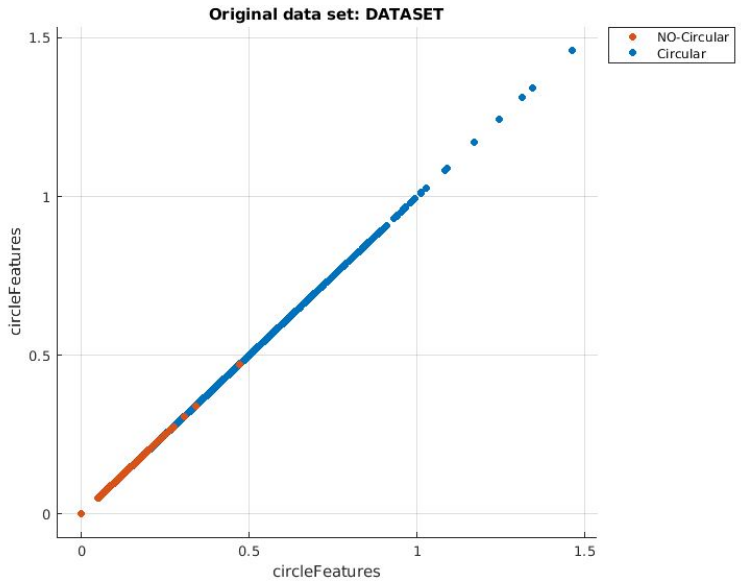


Figura 5. Scatter Plot

A la Figura 5 veiem la distribució obtinguda de les imatges. En aquest cas només tenim una dimensió, ja que estem utilitzant un únic valor que descriu la mètrica del cercle. A la gràfica podem apreciar una possible divisió de les dades en les dues classes.

El millor classificador trobat amb l'app *Classification Learner* ha sigut un arbre de decisions, on hem obtingut una precisió del 71.8%.

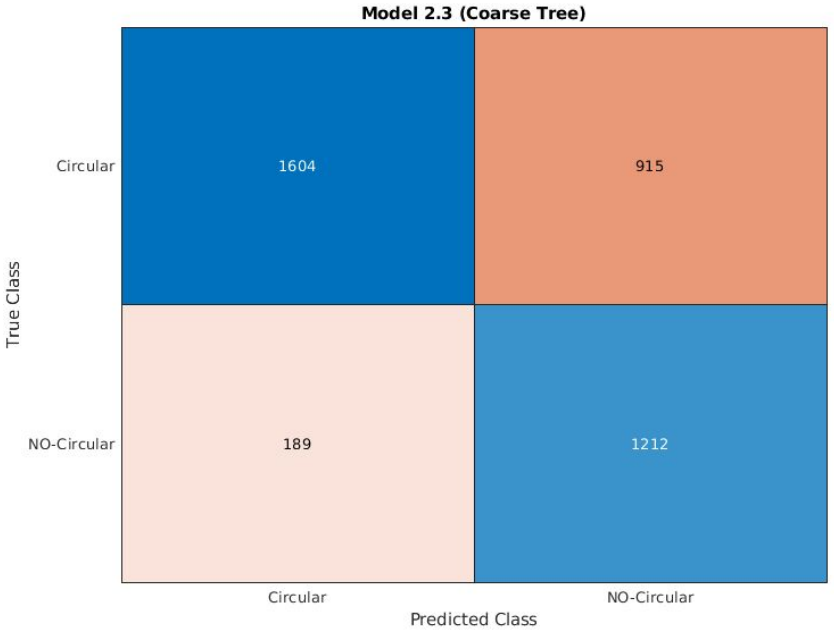


Figura 6. Confusion Matrix

La Figura 6 mostra la matriu de confusió obtinguda. Els resultats d'aquest descriptor són força positius, ja que permet classificar amb una precisió suficient bona i utilitzant només una variable, que permet reduir el temps de l'entrenament del classificador, i compensar l'elevat temps de la transformada de Hough.

Tot i els bons resultats, encarar podem veure un elevat nombre de senyals circulars classificats erròniament com a senyals no circulars. Per aquest motiu haurem d'utilitzar nous descriptors per obtenir millors resultats.

4.3 Histograma de gradients orientats (HOG)

El descriptor de HOG ens retorna un vector de característiques d'una llargada determinada per la mida de la imatge i la mida de la cel·la escollida. Com ja hem comentat en l'apartat anteriorment, primer modifiquem la mida de les imatges per a què totes siguin de 128x128, i

després obtenim el vector de característiques amb un tamany de cel·la de [16 16]. El resultat que obtenim és un vector de 1764 valors.

Primer de tot, per comprovar la millora d'aquest descriptor respecte el descriptor utilitzat anteriorment de la transformada de Hough circular, mirarem la precisió que obté per classificar senyals circulars de senyals no circulars.

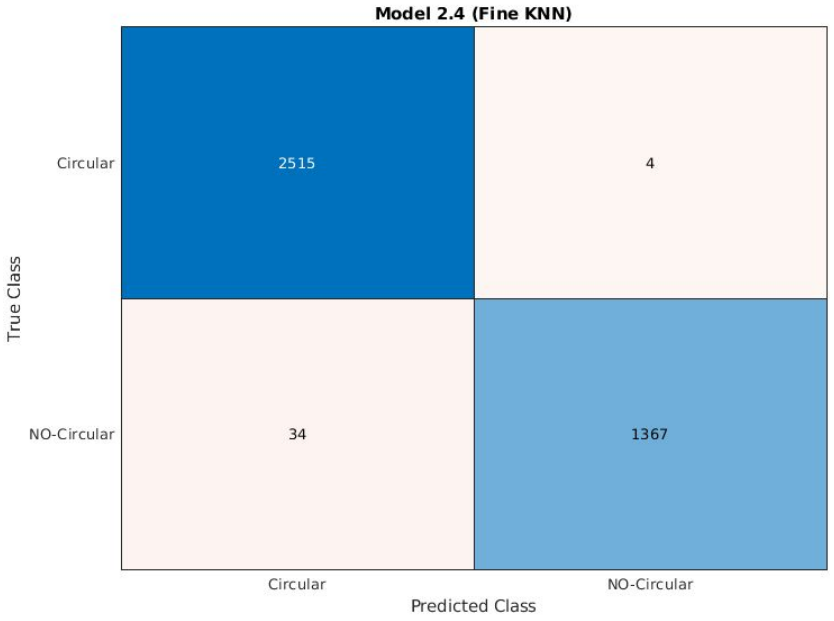


Figura 7. Confusion Matrix

Utilitzant el *classification learner APP*, obtenim una precisió del 99% amb el classificador Fine KNN. El resultat és molt més prometedor que l'obtingut amb la transformada de Hough, però s'ha de tenir en compte la quantitat de variables utilitzades i el temps de computació elevat que comporta tant obtenir el vector de característiques com entrenar els classificadors. I depenent de l'objectiu del nostre projecte (captura temps real), pot ser més o menys positiu.

Ara només hem mirat la precisió de dividir els senyals en dues subclases, però el nostre objectiu és poder classificar-les en grups individuals. Mirarem quina precisió podem obtenir amb els descriptors de HOG intentant classificar els senyals en les 5 formes que poden tenir.

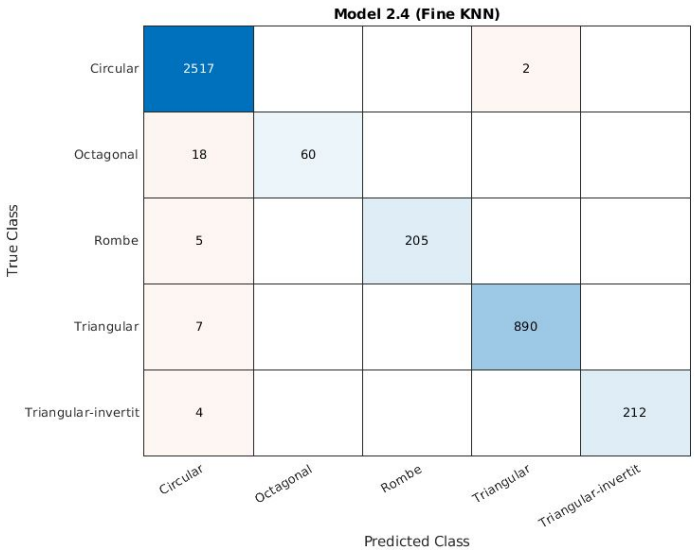


Figura 8. Confusion Matrix

El millor resultat obtingut ha sigut amb la mateixa precisió del 99% amb l'algorisme Fine KNN. A la Figura 8 veiem la matriu de confusió. Podem determinar que la classificació utilitzant HOG és molt positiva, però hem de tenir en compte l'elevat temps de computació a causa del nombre de variables.

4.4 Speeded Up Robust Feature (SURF)

Com a últim experiment abans de generar el classificador final, mirarem la precisió obtinguda amb el descriptor de SURF classificant els senyals en les 5 formes. El vector de característiques que obtenim és de 500 elements, bastant més petit que l'obtingut amb HOG.

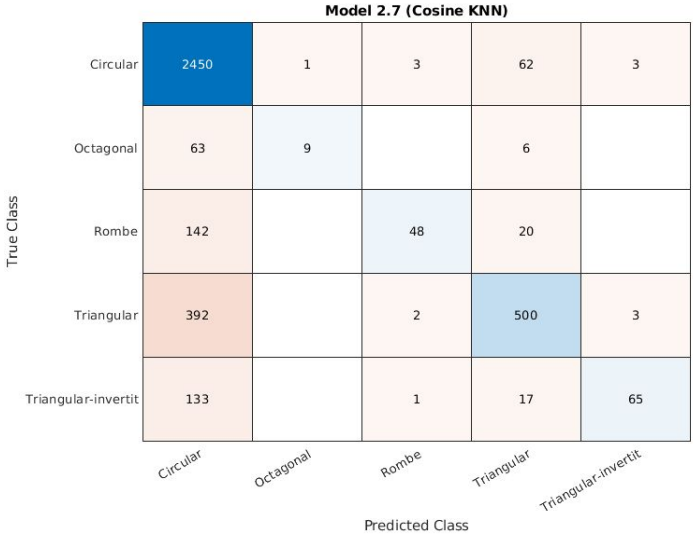


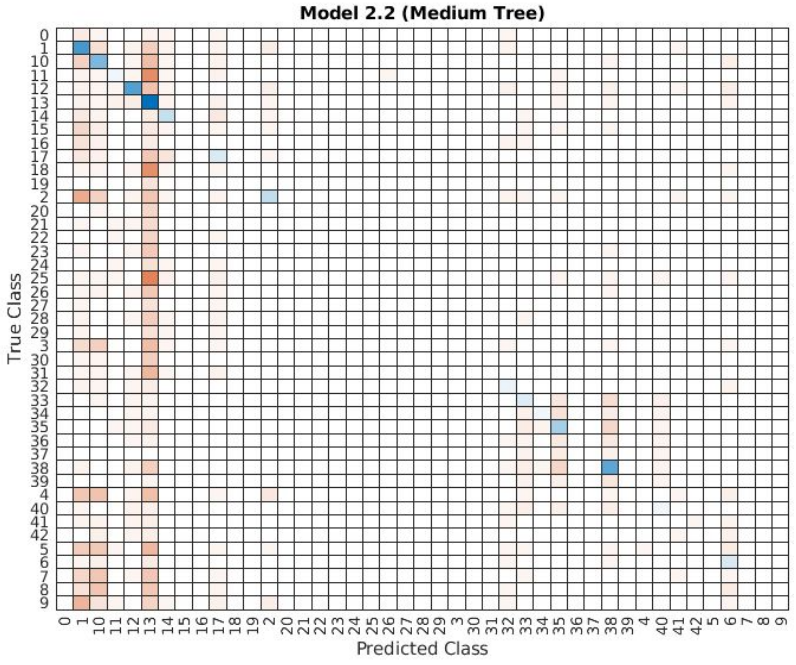
Figura 9. Confusion Matrix

La precisió obtinguda ha sigut d'un 78.4% amb l'algorisme Cosine KNN. El resultat, com es veu a la matriu de confusió, és bastant més dolent que l'obtingut amb HOG, però amb un temps de computació molt més petit, ja que només fem servir 500 variables per imatge.

5 Resultats

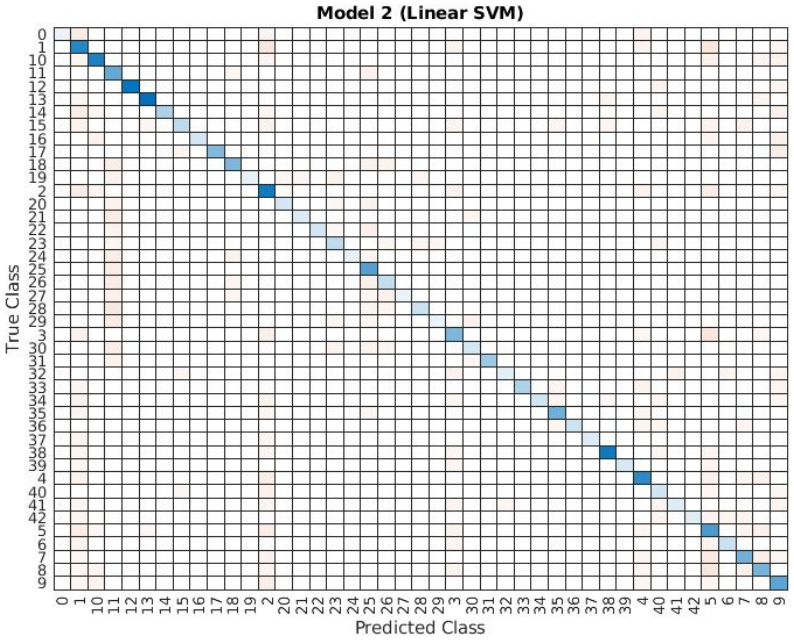
A continuació mostrem els resultats finals de classificar els 43 senyals utilitzant diferents conjunts de descriptors.

5.1 Color Feature - Hough Circle



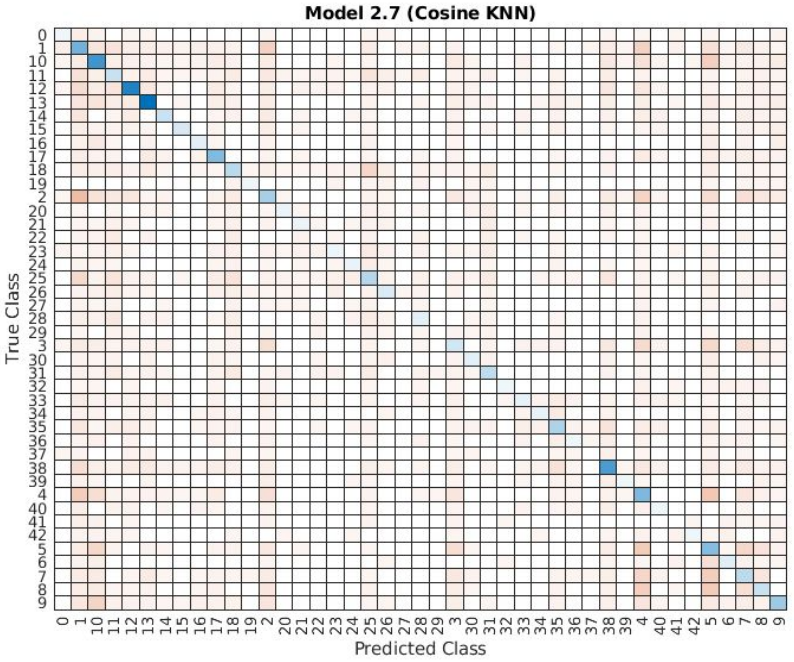
Precisió: 20.7% amb Medium Tree

5.2 Color Feature - HOG



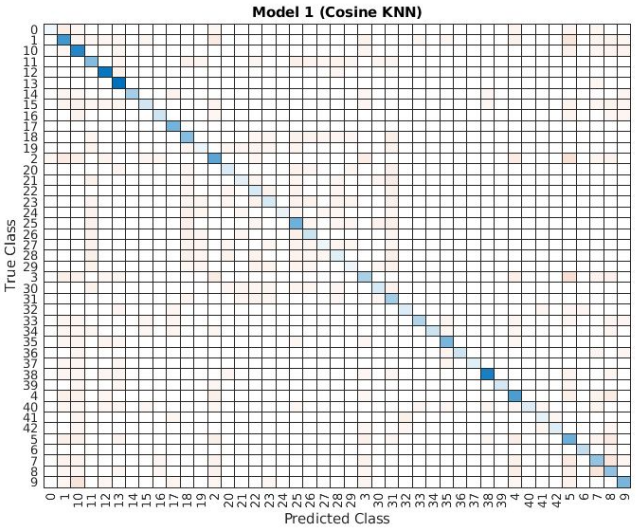
Precisió: 83.0.% amb Linear SVM

5.3 Color Feature - SURF



Precisió: 43.7% amb Cosine KNN

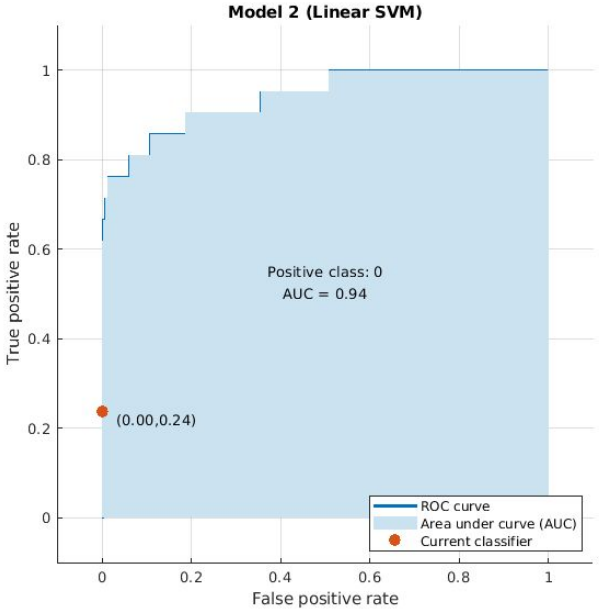
5.4 Color Feature - Hough Circle - HOG - SURF



Precisió: 82.1% amb Cosine KNN

5.5 Resultat Final

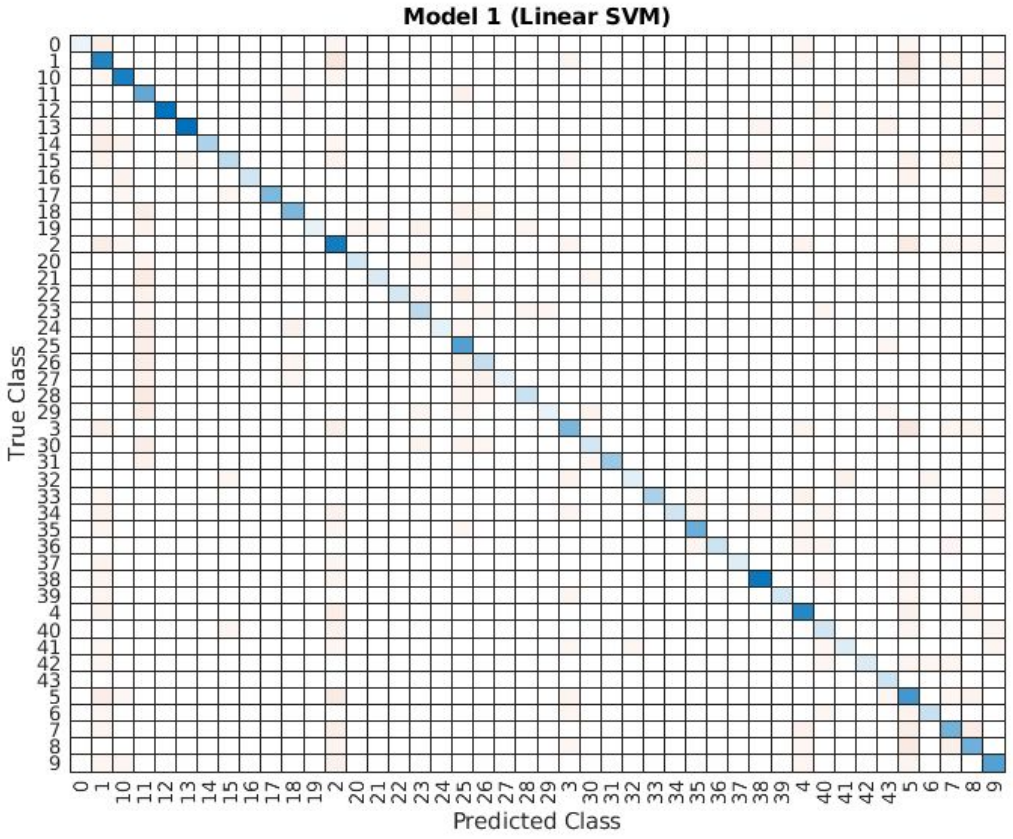
Concloem que la millor combinació de descriptors és la combinació de Color Feature i HOG, obtenint una precisió del 83.0% utilitzant l'algorisme Linear SVM en la classificació dels 43 senyals de trànsit. Quedem satisfets amb el resultat obtingut, però amb la intenció d'investigar millors descriptors per obtenir precisions més altes.



ROC Curve. Color Feature i Hog

6 Classe *'Reject'*

Hem afegit una nova classe, la qual serveix per rebutjar les imatges que no continguin cap mena de senyal de les que tenim per a identificar. Aquesta classe ha sigut entrenada amb un conjunt d'imatges extretes de *google images.* Principalment són imatges de carrers sense cap senyal de trànsit, ja que és l'escenari més probable del nostre projecte.



El resultat obtingut, afegint aquesta nova classe, és lleugerament més dolent que el resultat anterior. La màxima precisió obtinguda és del 78% amb Linear SVM. Aquest empitjorament és l'esperat en afegir l'extensió de 'Reject'. Creiem que ens faria falta un *dataset* amb més imatges per poder millor el resultat.

7 Annex: Codi

7.1 Descriptors

- **getColorFeature.m**

```
function [feature] = getColorFeature(image)
   I = image;
   hsv = rgb2hsv(I);
   h = hsv(:, :, 1);
   s = hsv(:, :, 2);
   v = hsv(:, :, 3);

   RED = (h < 15/360 | h > 345/360) & (s > 0.3);
   BLUE = (h > 200/360 & h < 230/360) & (s > 0.5);
   YELLOW = (h > 25/360 & h < 50/360) & (s > 0.5);
   BLACK = (v < 0.15);

   numPixel = numel(I(:,:,1));

   pRed = sum(RED(:) / numPixel);
   pBlue = sum(BLUE(:) / numPixel);
   pYellow = sum(YELLOW(:) / numPixel);
   pBlack = sum(BLACK(:) / numPixel);

   feature = [pRed pBlue pYellow pBlack];
end
```

- **getHoughFeature.m**

```
function [metricStrong] = getHoughFeature(image)
   I = image;

   [rows, columns] = size(I(:,:,1));
   max_radi = floor(min(rows, columns)/2);
   min_radi = max(floor(max_radi/3), 6);

   center_x = columns/2;
   center_y = rows/2;

   [centers, radii, metric] = imfindcircles(I,[min_radi
max_radi], 'Sensitivity', 0.95);

   metricStrong = 0;
   for i = 1:size(centers, 1)
       dist_x = abs(centers(i,1) - center_x);
       dist_y = abs(centers(i,2) - center_y);

       if dist_x < columns/5 && dist_y < rows/5
           metricStrong = metric(i);
           break
       end
   end

end
```

- **getHogFeature.m**

```
function [hog_vector] = getHogFeature(img)
    [rows, columns] = size(img(:,:,1));
    if rows > columns
        dif = rows - columns;
        img = imcrop(img, [0, dif/2, columns, columns-1]);
    elseif columns > rows
        dif = columns - rows;
        img = imcrop(img, [dif/2, 0, rows-1, rows]);
    end

    img = imresize(img, [128, 128]);
    hog_vector = extractHOGFeatures(img,'CellSize',[16 16]);
end
```

- **SURF: Es obtingut a getClassifier.m**

7.2 Training

- **getClassifier.m**

```
colorFeatures = [];
hogFeatures = [];

% houghFeatures = [];
% surfFeatures = [];

D = '~/Documents/MATLAB/Traffic_images/Train-0-15-29/'

setDir  = fullfile(D);
imds =
imageDatastore(setDir,'IncludeSubfolders',true,'LabelSource'
,'foldernames');
% bag = bagOfFeatures(imds);

while hasdata(imds)
   img = read(imds);
   colorFeatures = [colorFeatures; getColorFeature(img)];
   hogFeatures = [hogFeatures; getHogFeature(img)];
   % houghFeatures = [houghFeatures; getHoughFeature(img)];
   % surfFeatures = [surfFeatures; encode(bag, img)];
end

DATASET = [colorFeatures hogFeatures];
classifier = fitcecoc(DATASET, imds.Labels);
save('traffic_sign_classifier.mat', 'classifier');
```

7.3 Predict

- **classify_traffic_sign.m**

```
function [class] = classify_traffic_sign(img)

   load('traffic_sign_classifier.mat');

   features = [getColorFeature(img) getHogFeature(img)];
   class = predict(classifier, features);

   subplot(1,2,1);
   imshow(img);
   title('Possible Traffic Sign')

   % imatge de Metadata
   subplot(1,2,2);
   detected = imread(join( [char(class), '.png'] ) );
   imshow(detected);
   title('Detected Traffic Sign')

end
```