Classification d'images télévisées

Farhad - Heybati
Youcef - Kacer
Martin - Provost

9 May 2016

Table des matières

Introduction

Ce document présente plusieurs méthodes d'extraction d'informations à partir d'images issues d'un débat télévisé. Cela afin d'en effectuer une classification binaire en « gros plan » ou « large plan ». Nous proposons dans un premier chapître de présenter les images exploitées, et les deux classes qui nous intéressent. Dans une seconde partie, nous presentons les méthodes d'extraction d'informations utilisées. Puis dans une troisième partie, les résultats obtenus.

Chapitre 1

Images exploitées

1.1 Corpus et labelisation

Nous allons exploités un total de 2351 images prises à partir de la vidéo d'un débat télévisé [Ess16] . Nous avons exploité le fichier de transcription au format .trs [Ess16] associé à cette vidéo, cela afin d'extraire les différents classes, et attribuer à chaque image sa classe. En effet, le fichier de transcription fournit un total de 9 classes :

M **:** La présentatrice est seule à l'écran

A **:** La première intervenante est seule à l'écran

B **:** La seconde intervenante est seule à l'écran

C **:** Le premier intervenant est seul à l'écran

D **:** Le second intervenant est seul à l'écran

ALL **:** Les 5 personnes sont à l'écran

$MULTI$ **:** Entre 2 et 4 personnes sont à l'écran

$INTRO$ **:** Reportage d'introduction à l'écran

$CREDITS$ **:** Générique d'emission à l'écran

Par ailleurs, le fichier de transcription donne à chaque intervalle de temps, ce qui est à l'écran parmi les classes citées plus-haut. Moyennant un reformatage des données xml de ce fichier, on peut obtenir le tableau suivant :

classe	**debut (s)**	**fin (s)**
$CREDITS$	0	11.36
$INTRO$	11.36	84.64
M	84.64	95.12
ALL	95.12	103.2
A	103.2	112.2
M	112.2	115.24
$\vdots$	$\vdots$	$\vdots$
M	2334.12	2340
ALL	2340	2340.76
$CREDITS$	2340.76	2349.72

TABLE 1.1 – Table de correspondance classes/intervalle de temps

Les 2351 images étant prises à une seconde d'intervalle tout le long de la vidéo, on peut automatiquement labéliser celles ci via la table de correspondance 1.1. Ci-après, nous présentons quelques images pour chacune des 9 classes (Cf. figures 1.1,1.2,1.3,1.4,1.5, 1.6,1.7,1.8,1.9).

1.2 Classification binaire

Par la suite, nous allons nous resteindre à seulement deux classes définies comme suit :

G **:** « Gros plan » (une seule personne est à l'ecran)
L **:** « Large plan » (au moins deux personnes sont à l'écran)

Les classes G et L peuvent s'exprimer en fonction des 9 classes comme suit :

G **:** $M \mid A \mid B \mid C \mid D$
L **:** $ALL \mid MULTI$

Nous avons donc deux classes d'images G et L (respectivement 1491 et 761 images) (Cf. figures 1.10 et 1.11)

Par la suite, nous allons expliciter différentes manières d'extraire de l'information afin de pouvoir discriminer les images de classe G , des images de classe L.

FIGURE 1.1 – image de classe M

FIGURE 1.2 – image de classe A

FIGURE 1.3 – image de classe B

FIGURE 1.4 – image de classe C

FIGURE 1.5 – image de classe D

FIGURE 1.6 – image de classe ALL

FIGURE 1.7 – image de classe $MULTI$

FIGURE 1.8 – image de classe $INTRO$

FIGURE 1.9 – image de classe *CREDITS*

FIGURE 1.10 – image de classe G (gros plan)

FIGURE 1.11 – image de classe L (large plan)

Chapitre 2

Extraction d'information

2.1 Extraction de contours

Cette méthode consiste à déterminer le nombre de contours détecté par des algorithmes de traitement d'image contenu dans *OpenCV*[Bra00], et d'en déduire si l'image est un « gros plan » ou un « large plan ».
Nous effectuons 2 passes sur chaque image avec des paramétrages d'algorithmes différents afin de produire 2 variables par image.
Dans chaque passe nous ajoutons un flou Gaussien à l'image afin d'atténuer les petits détails comme les traits du visage ou du décor.
Ensuite, l'image est passée en niveau de gris pour être binarisée via un seuil adaptative (cv :: AdaptativeThreshold)
Et enfin, on execute la fonction (cv :: findContours) avec une option différente pour chaque passe.

La première passe extrait les contours avec l'option *CHAIN_ APPRO_ NONE*.
La seconde passe extrait les contours avec l'option *CHAIN_ APPROX_ SIMPLE*.

Les 2 informations du nombre de contours sont stockées ensuite dans un DataFrame comme descripteur destiné à être classifié de manière supervisée.

2.2 Extraction de saturation

2.2.1 Définition

Hue, Saturation, Value ou HSV est un modèle qui représente la couleur (H) en fonction de la saturation (S) en gris et la luminosité (V) (Cf. figure 2.1).

Teinte (Hue) : Correspond à la couleur. Elle est définie sous la forme d'un cercle où les trois couleurs primaires RVB sont réparties aux angles 0 ° pour le rouge, 120 ° pour le vert et 240 ° pour le bleu.

Saturation : C'est le taux de pureté de la couleur, qui varie entre un maximale (couleur éclatante) et l'achromatisme (couleur gris).

Valeur : C'est la mesure de l'intensité lumineuse de la couleur qui varie entre le noir et le blanc.

2.2.2 Choix de la variable explicative

Nous sommes partis de l'hypothèse que les images en « gros plan » sont plutôt gris et donc une saturation en couleur plus faible que les images en « large plan ». La réalisation d'un histogramme sur la moyenne de saturation (Cf. Figure 2.2), portant sur la totalité des images, nous a permis d'identifier deux classes de type Gaussien pour modéliser nos 2 classes qui sont les images de type « gros plan » et les images de type « large plan »
Nous présentons le code Python d'extraction de saturation et sa classification dans l'annexe A

La moyenne de la saturation globale obtenue est 60.2561640854. Nous avons considérés les images dont la saturation moyenne est inférieure à la moyenne globale comme appartenant à la classe « gros plan » et les images dont la saturation moyenne est supérieure à la moyenne globale comme appartenant à la classe « large plan ».

2.3 Histogramme orienté du gradient

Cette méthode consiste à extraire l'information local de contours. Historiquement, sa première utilisation a consisté en la détection de piétons [DT05]. Pour une image donnée, le vecteur descripteur des *HOG* est la concaténation d'histogrammes de l'amplitude du gradient (en fonction de son orientation). Chaque histogramme est pris sur un carré de l'image appelé « cellule ». L'ensemble des cellules quadrille entièrement l'image avec un eventuel recouvrement.
Ces histogrammes sont ensuite normalisés en intensité par paquet de cellules adjacentes (appelés « block ») afin d'obtenir des descripteurs invariants au changement local d'illumination (Cf. figure 2.3).

On peut espérer que ces descripteurs s'adaptent bien à notre problème. En effet, les epaules des intervenants à gauche et à droite de l'image, ainsi que leur visage, sont des contours descriminants pour la classe G (gros plan) (Cf. figure 2.4). D'autre part, les gros plans ont un fond uniforme, ce qui donnera beaucoup de gradient nul, et donc un descripteur *sparse* pour la classe G (gros plan).

2.4 Reseaux de neurones à convolution pré-entrainé

Cette méthode consiste à extraire les descripteurs produits par un réseau de neurones à convolution déjà entrainé. En effet, en récupérant la sortie de l'avant-dernière couche, on obtient la transformation finale haut-niveau de l'image, avant la couche de classification. Cette technique est connue sous le nom de *Transfer Learning* [YCBL14] et a fait ses preuves dans le cas de réseaux de neurones entrainé sur la base d'images *ImageNet* [DDS$^+$09]. Cette base de près de 10 millions d'images, contient une multitude de classes (animaux,ustensiles,humains,...)

organisées hiérarchiquement, et on peut espérer que les gros plans (classe G) correspondent à l'une de ces classes,et de même pour les images plan large (classe L), de telle sorte qu'on puisse ensuite séparer les descripteurs via un classifieur supervisé.

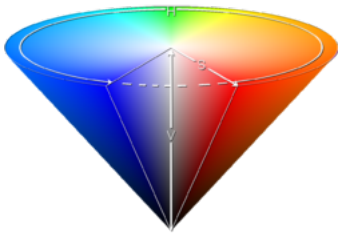

FIGURE 2.1 – Illustration HSV [Wik16]

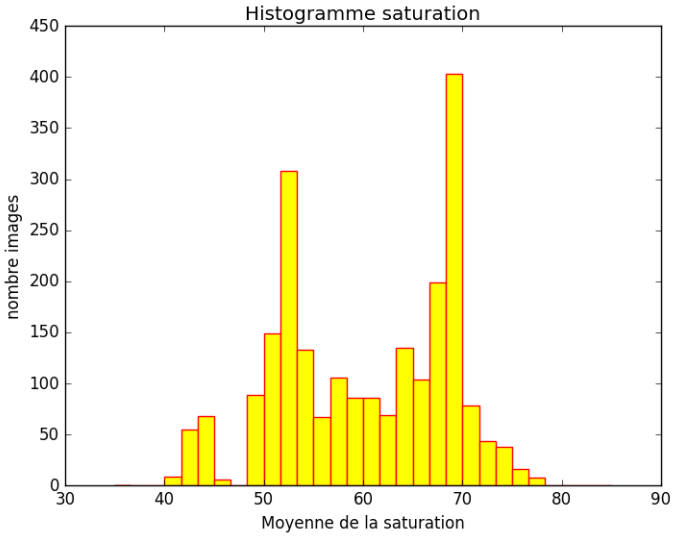

FIGURE 2.2 – Histogramme des moyennes de saturation

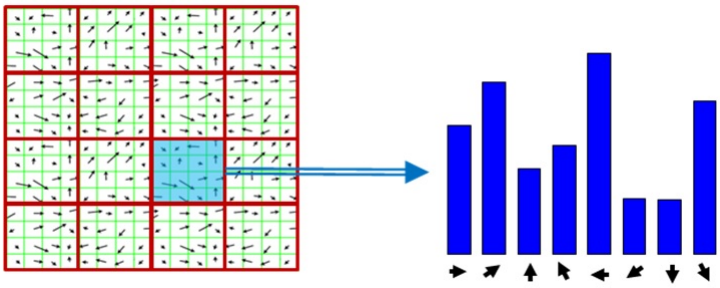

FIGURE 2.3 – Construction d'un histogramme orienté du gradient [Lev13]

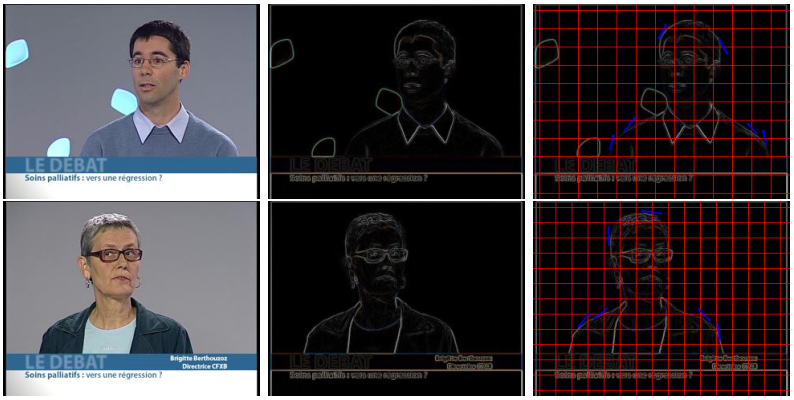

FIGURE 2.4 – Histogramme orienté du gradient pour les gros plans (classe G)

Chapitre 3

Résultats

3.1 Extraction de contours

Pour expliquer le résultat de l'extraction de contours, voici quelques exemples d'images où les contours ont été tracés avec la fonction cv :: drawContours.

La figure 3.1 montrent deux exemples de contouring sur des images « gros plan » (classe G), appliqué avec l'option *CHAIN_ APPRO_ NONE* (première passe)

La figure 3.2 montrent deux exemples de contouring sur des images « large plan » (classe L), appliqué avec l'option *CHAIN_ APPRO_ NONE* (première passe)

Il y a plus de contours dans les images « large plan » car le décor du plateau TV est composé de beaucoup de détails. Cela permet de bien différencier les gros plans des plans larges mais pour quelques cas le nombre de contours ne suffit pas.
Pour remédier à cela, nous avons mis en place la seconde passe et dont voici les résultats pour les mêmes images :

La figure 3.3 montre deux exemples de contouring sur des images « gros plan » (classe G), appliqué avec l'option *CHAIN_ APPRO_ SIMPLE* (seconde passe).

La figure 3.4 montre deux exemples de contouring sur des images « large plan » (classe L), appliqué avec l'option *CHAIN_ APPRO_ SIMPLE*

Le nombre de contours à beaucoup augmenté par rapport à la première passe, ce qui va permettre de différencier encore mieux les types de plans : en effet, on note ici un très grand nombre de contours dans le décor et sur les personnages, ce qui nous permet de mieux discriminer les plans larges. Le bandeau du bas est plus détaillé en termes de contours que pour la première passe.

classifieur	**score de test**	**temps de training (s)**	**temps de test (s)**
Ridge Classifier	0.67	0.44	0.001
Perceptron	0.652	0.005	0.0
Passive-Aggressive	0.344	0.005	0.0
kNN	0.779	0.002	0.003
Random forest	0.906	0.421	0.021
linear SVM - L2 penalty	0.630	0.002	0.0
SGDClassifier - L2 penalty	0.655	0.003	0.0
linear SVM - L1 penalty	0.786	0.007	0.0
SGDClassifier - L1 penalty	0.656	0.005	0.0
SGDClassifier - ElasticNet penalty	0.616	0.005	0.0
NearestCentroid	0.525	0.001	0.0
SVM	0.902	0.215	0.062
linear SVM - L1 penalty - dual=False	0.670	0.067	0.0

TABLE 3.1 – Résultats de classification supervisée sur les moyennes de saturation

3.2 Extraction de teinte

En prenant comme frontière de décision, la valeur calculée en 2.2.2, alors on obtient un taux de prédiction correcte de 78.32%.
Par ailleurs, nous avons demandé à différent classifieurs d'effectuer un apprentissage supervisé sur les moyennes de saturation. Pour cela, nous avons divisé les images en deux sous-ensembles représentant 70% du total pour l'entrainement, 30% pour le test.
Les resultats de cette classification sont illustrés dans la table 3.1. On voit que SVM et Random Forest donnent des frontières plus complexes qui permettent un meilleure score de test (90%).
Les classifieurs utilisés sont ceux de la librairie *Scikit-Learn*, implémentés dans le script *Python* tiré de [PGBB14], que nous avons adaptés.

3.3 Histogramme orienté du gradient

Nous avons extrait les descripteurs *HOG* pour chacune des images de classe G et L, en utilisant une cellule de taille 32x32 (sans recouvrement) de sorte à récupérer les formes des épaules et du visage, et non les details plus petits liés au port de lunettes par exemple. Pour la normalisation, nous l'avons effectué par groupe de carré à 4 cellules (avec recouvrement) afin de tenir compte du fait qu'au sein des gros plans, les intervenants peuvent être légèrement décalés. Puis, nous avons appliquer à ces descripteurs, différent classifieurs supervisés. Nous

classifieur	**score de test**	**temps de training (s)**	**temps de test (s)**
Ridge Classifier	1.0	0.130	0.027
Perceptron	1.0	0.219	0.028
Passive-Aggressive	1.0	0.352	0.024
kNN	0.998	0.221	2.875
Random forest	1.0	1.321	0.012
linear SVM - L2 penalty	1.0	0.197	0.024
SGDClassifier - L2 penalty	1.0	0.202	0.024
linear SVM - L1 penalty	1.0	0.283	0.024
SGDClassifier - L1 penalty	1.0	0.634	0.024
SGDClassifier - ElasticNet penalty	1.0	0.721	0.024
NearestCentroid	0.996	0.049	0.028
SVM	0.998	0.446	0.203
linear SVM - L1 penalty - dual=False	1.0	0.238	0.003

TABLE 3.2 – Résultats de classification supervisée sur les descripteurs *HOG*

avons découpé le set de descripteurs en deux sous-ensembles représentant 70% du total pour l'entrainement, 30% pour le test. Les resultats de classification sont illustrés dans la table 3.2.
Les classifieurs utilisés sont ceux de la librairie *Scikit-Learn*, implémentés dans le script *Python* tiré de [PGBB14], que nous avons adaptés.

3.4 Reseaux de neurones à convolution pré-entrainé

Nous avons utilisé le code *Overfeat* [SEZ+13], récupéré par clonage du dépôt Github correspondant [SEZ+14], pré-entrainé sur la base *ImageNet* [DDS+09]. Nous avons extrait les descripteurs pour chacune des images de classe G et L pour leur appliquer différent classifieurs supervisés. Nous avons découpé le set de descripteurs en deux sous-ensembles 70% du total pour l'entrainement, 30% pour le test. Les resultats de classification sont illustrés dans la table 3.3.
Les classifieurs utilisés sont ceux de la librairie *Scikit-Learn*, implémentés dans le script *Python* tiré de [PGBB14], que nous avons adaptés.

classifieur	**score de test**	**temps de training (s)**	**temps de test (s)**
Ridge Classifier	1.0	0.762	0.152
Perceptron	0.998	1.515	0.155
Passive-Aggressive	1.0	1.852	0.150
kNN	0.998	1.456	16.77
Random forest	0.998	3.851	0.024
linear SVM - L2 penalty	1.0	1.105	0.149
SGDClassifier - L2 penalty	0.998	1.156	0.153
linear SVM - L1 penalty	1.0	1.220	0.149
SGDClassifier - L1 penalty	1.0	5.581	0.149
SGDClassifier - ElasticNet penalty	1.0	6.062	0.151
NearestCentroid	0.987	0.302	0.221
SVM	1.0	3.508	1.591
linear SVM - L1 penalty - dual=False	1.0	1.150	0.011

TABLE 3.3 – Résultats de classification supervisée sur les descripteurs $Overfeat$

FIGURE 3.1 – Contouring sur des images de classe G avec l'option CHAIN_APPRO_NONE

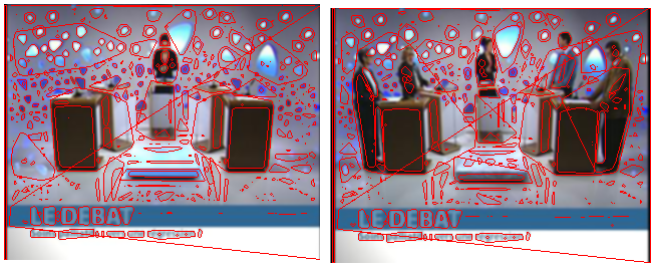

FIGURE 3.2 – Contouring sur des images de classe L avec l'option CHAIN_APPRO_NONE

FIGURE 3.3 – Contouring sur des images de classe G avec l'option CHAIN_APPRO_SIMPLE

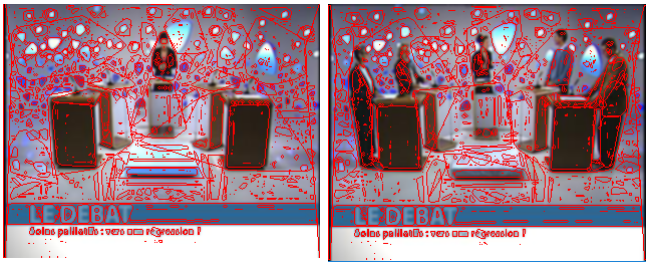

FIGURE 3.4 – Contouring sur des images de classe L avec l'option CHAIN_APPRO_SIMPLE

Annexe A

Code Python pour l'extraction de saturation et sa classification

```
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt
import shutil
import pandas as pd

saturation = []
path='c:/tmp/images_debat/'
dirs = os.listdir(path)
file_valid='c:/tmp/labels.csv'
# This would print all the files and directories
for file in dirs:
    if file.endswith((".jpg")):
        colorImage = cv2.imread(path+file)
        #print ('Traitement du fichier : ',path+file)
        rows, cols, nbChannels = colorImage.shape

        b,g,r = cv2.split(colorImage)
        colorImage=cv2.merge([r,g,b])
        color = ('b','g','r')
        clorImageHSV=cv2.cvtColor(colorImage, cv2.COLOR_BGR2HSV)
        h,s,v=cv2.split(clorImageHSV)
        sm=np.mean(s)
        saturation.append(sm)
        #if file == '00000108.jpg' or file == '00000110.jpg' or file == '00000147.jpg':
```

```
            #print ('saturation pour le fichier 108 - 110 -147:')
            #print sm

globalMean = np.mean(saturation)
print ('Moyenne globale:')
print globalMean
#print saturation
plt.hist(saturation, range = (35, 85), bins = 30, color = 'yellow',
            edgecolor = 'red')
plt.xlabel('Moyenne de la saturation')
plt.ylabel('nombre images')
plt.title('Histogramme saturation')
#hist=np.histogram(saturation)
#plt.bar(hist)
plt.show()
copyPathGrosPlan='c:/tmp/GrosPlan/'
copyPathLargePlan='c:/tmp/LargePlan/'
classif=pd.DataFrame(columns=('file_name', 'label'))
classif_multi=pd.DataFrame(columns=('file_name', 'sm'))
line = 0;
for file1 in dirs:
    if file1.endswith((".jpg")):
        colorImage = cv2.imread(path+file1)
        rows, cols, nbChannels = colorImage.shape

        r,g,b = cv2.split(colorImage)
        colorImage=cv2.merge([r,g,b])
        color = ('b','g','r')
        clorImageHSV=cv2.cvtColor(colorImage, cv2.COLOR_BGR2HSV)
        h,s,v=cv2.split(clorImageHSV)
        sm=np.mean(s)
        classif_multi.loc[line]=(file1,sm)
        #Gros plan
        if  sm <= (globalMean):
             classif.loc[line]=(file1,-1.)
        #Large plan
        else:
            classif.loc[line]=(file1,1.)

        line = line+1

classif.sort_values(['file_name'])
#print classif
#Open the labelled file
labelfile = open(file_valid)
label_img = pd.read_csv(labelfile,sep=';', names=['file_name', 'label'],header=0,dtype=
```

```
label_img.sort_values(['file_name'])
#print label_img

correct_predict = pd.merge(classif, label_img, on=['file_name', 'label'], how='inner')
#print len(correct_predict)
#print len(label_img)
print ('Resultat :', (float(len(correct_predict))/float(len(label_img)))*100.)
#print classif_multi
#print label_img
label_img_mrg = pd.merge(classif_multi, label_img, on=['file_name'], how='inner')
label_img_mrg = label_img_mrg.drop('file_name', 1)
#print label_img_mrg
label_img_mrg.to_csv('C:\perso\TP\Son-Image\classif_multi.csv', sep=';', encoding='utf-8')
```

Liste des tableaux

Table des figures

Bibliographie

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet : A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *cvpr*, jun 2005.

[Ess16] Slim Essid. Resources, 2016.

[Lev13] Gil Levi. A short introduction to descriptors, aug 2013.

[PGBB14] Peter Prettenhofer, Olivier Grisel, Mathieu Blondel, and Lars Buitinck. Classification of text documents using sparse features, 2014.

[SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat : Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[SEZ+14] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat code source, 2014.

[Wik16] Wikipédia. Teinte saturation valeur — wikipédia, l'encyclopédie libre, 2016. [En ligne ; Page disponible le 30-avril-2016].

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks ? *CoRR*, abs/1411.1792, 2014.