Mémoire de Projet de Fin de 2ème Année

Sujet

Reconnaissance des expressions faciales

Soutenu par :

Mohamed BAMOUH

Adam SADDIKI

Sous la direction de :

Mme. Sanaa ELFKIHI

Année Universitaire 2017-2018

Table des figures

Sommaire

Remerciements :

Nous tenons à remercier profondément tous nos professeurs de l'ENSIAS, qui nous ont formés et accompagnés tout au long de ces deux années avec beaucoup de patience et de pédagogie, et qui, grâce à eux, nous avons pu acquérir les connaissances théoriques ainsi que pratiques nous ayant permis de réaliser notre projet.

Nous remercions également nos proches et nos amis pour leur soutien accordé durant la période de réalisation de ce projet.

Et pour finir, notre gratitude est particulièrement destinée à Mme ElFkihi Sanaa pour avoir enrichi nos connaissances, son encadrement, son suivi continu de tout notre travail ainsi que tous ses efforts fournis.

Introduction :

Les émotions sont un facteur important dans la communication entre les êtres humains et nous aident à bien comprendre l'autre. Généralement, les gens peuvent exprimer leurs émotions selon 2 méthodes : La voix et/ou le visage. Comme la communication non verbale porte plus d'informations que la communication verbale, la plupart des études se concentre sur l'étude du visage plus que la voix.

L'être humain peut exprimer plusieurs émotions, mais les théoriciens propose qu'il y a que 6 émotions de base, les autres ne sont qu'un mélange de ces 6 émotions et qui sont : La joie, la tristesse, la peur, la colère, le dégoût et la surprise.

La reconnaissance des expressions faciales (Facial Emotion Recognition FER) est un sujet important dans le traitement des images, développement des techniques de l'intelligence artificielle, Interaction Homme-Machine, Création des réalités virtuelles ...

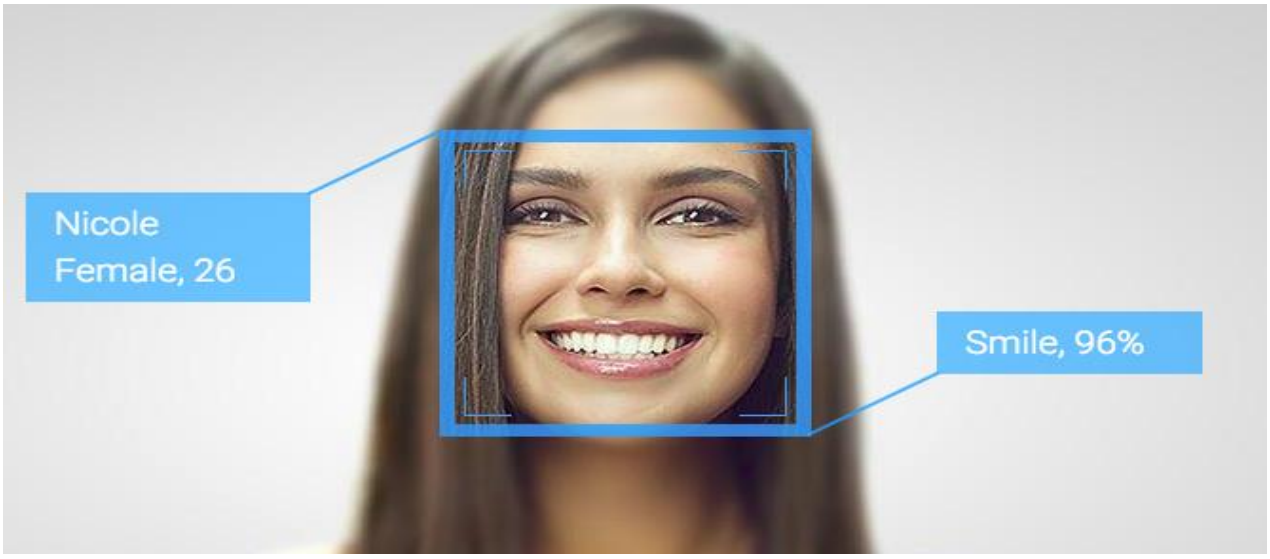


Figure 1: Exemple de l'utilisation du programme Face++ Congnitive Services

Plusieurs recherches ont été faites sur ce sujet et par conséquence l'existence de plusieurs algorithmes de détection des expressions faciales mais qu'on n'arrive pas à faire une comparaison entre eux, car chaque algorithme se base sur des calculs et des méthodes différentes que l'autre. Dans ce rapport on va essayer d'expliquer un algorithme de reconnaissance des expressions faciales et de créer un programme de reconnaissance en Opencv c++.

Chapitre I

Détection des visages

I- Détection des visages :

La première étape à suivre dans la détection des expressions faciales c'est la détection de visage qu'on arrive à réaliser en utilisant plusieurs méthodes :

Détection d'un visage dans une image avec un arrière-plan "monocolore" :

Comme indiqué dans le nom et c'est la méthode la plus simple dans la détection du visage, l'arrière-plan de l'image contient une seule couleur qu'on peut supprimer pour obtenir le résultat comme l'image dans l'exemple suivant :

Figure 2: Image contenant un visage dans un arrière-plan simple

Néanmoins, cela implique que l'image doit déjà contenir au moins un visage et que l'arrière-plan soit assez simple pour être supprimé après l'application d'un filtre.

En essence, cet algorithme est le plus simple en matière de détection de visages, et ne sert qu'à isoler la partie « visage » pour faire des traitements a part dessus.

Détection du visage en utilisant Opencv (Méthode Haar Cascade) :

Aspect théorique :

La méthode Haar Cascade, bien qu'elle soit appliquée à la détection de visages dans notre étude, peut servir à la reconnaissance de n'importe quel type d'objet, en effet, le principe de base de cette méthode repose sur l'utilisation d'une multitude de « weak classifiers », entrainés grâce à une multitude d'images

positives ou l'objet que l'on souhaite détecter est présent, et des images négatives ou l'objet n'existe pas, afin de créer un « strong classifier » qui détectera l'objet voulu avec précision.

L'algorithme repose sur l'utilisation de caractéristiques pseudo-Haar (Haar-features) bien spécifiques, dont voici quelques exemples :

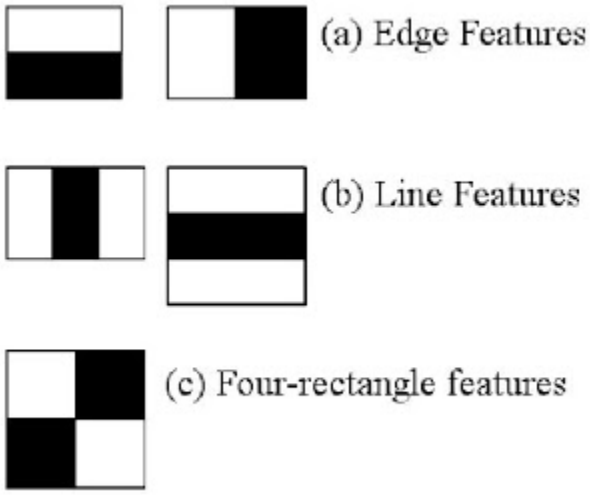


Figure 3: Quelques features utilisés par la méthode Haar Cascade

Figure 4 : Image en noir et blanc représentant le visage d'un homme

On peut distinguer quelques Haar-features dans cette image, par exemple,

- La zone des sourcils est claire au-dessus, et sombre en dessous, (edge feature)
- La zone autour du nez est sombre, tandis que le nez est plus clair, (line feature)

- Les dents sont claires et les lèvres sont plus foncées.

L'algorithme, après évidemment avoir converti l'image couleur en nuances de gris, traite ces zones-là. En soustrayant la somme des pixels dans la zone sombre a la somme des pixels dans la zone claire, on obtient une valeur Delta qui, comparée à la même somme appliquée sur une des Haar-features, permet de reconnaitre une partie du visage.

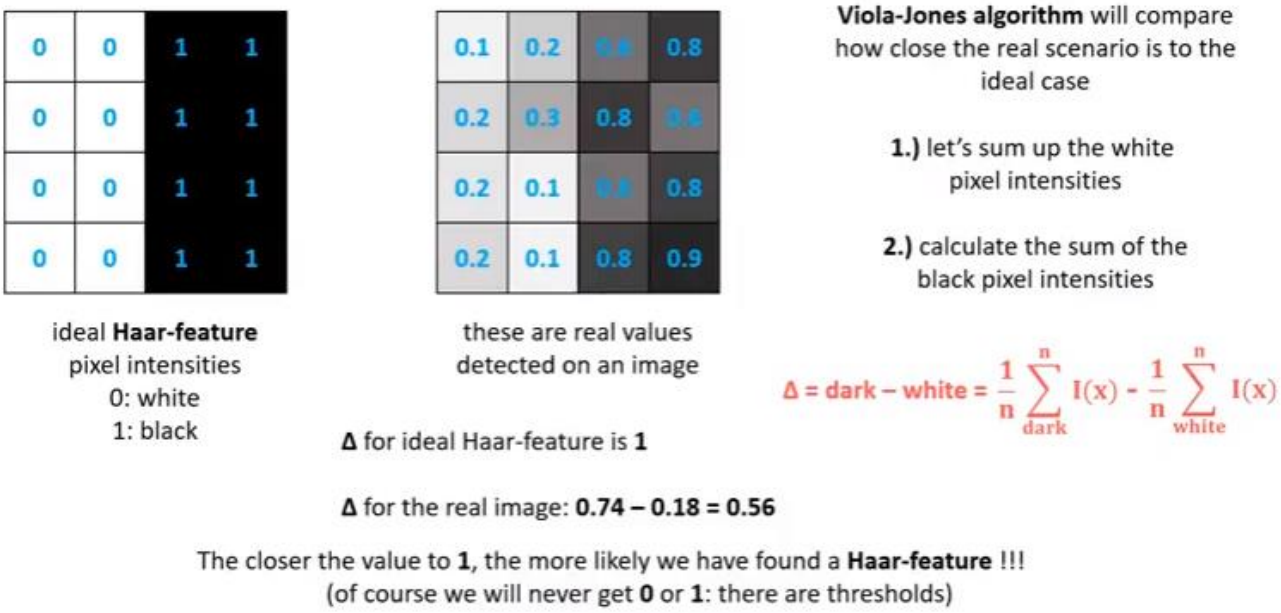


Figure 5: Algorithme de reconnaissance d'une Haar-feature sur une partie de l'image

Ce qui donne le résultat suivant :

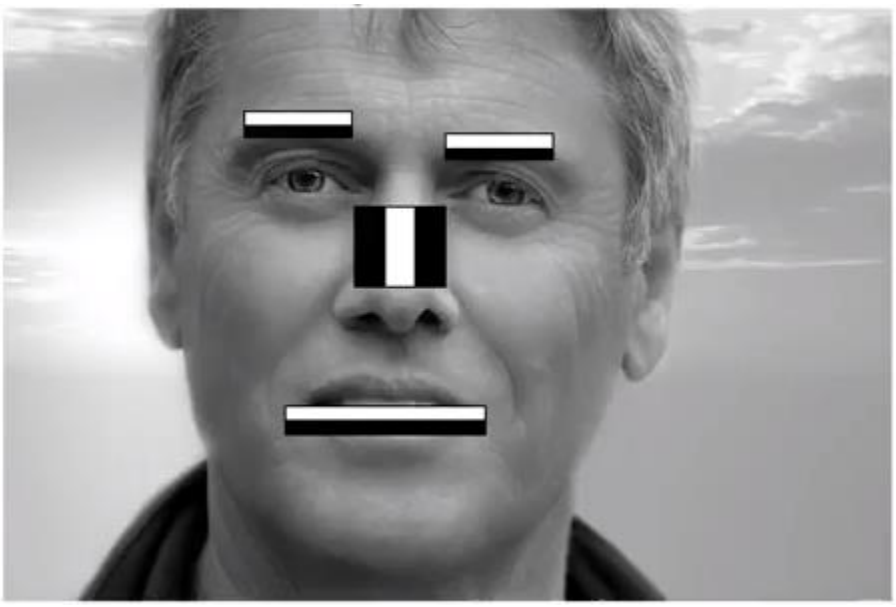


Figure 6: figure précédente mais avec application des Haar-features

L’algorithme applique plus de 6000 features dans chaque portion 24 x 24 d’une image, ce qui est une immense coté calculs et très détrimental coté performance, c’est pourquoi une image passe par une cascade de classifiers, si une portion de l’image rate une étape intermédiaire, le processus est interrompu et on passe à la portion suivante.

Pratique :

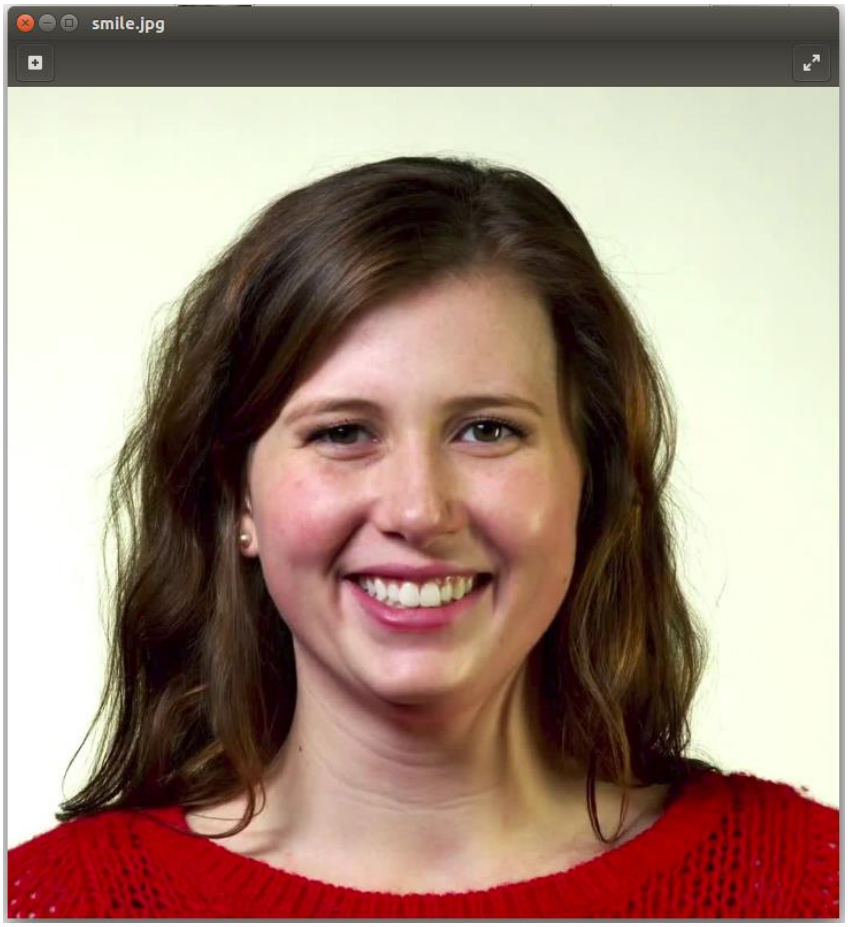


Figure 7: Image source

On définit la méthode pour détecter et afficher un visage dans un image :

- Transformer l'image source en une image au niveau de gris

  cvtColor(frame, frame_gray, *COLOR_BGR2GRAY*);

- Détection de visage en utilisant la méthode cascade :

  face_cascade.**detectMultiScale**(frame_gray, faces, 1.1, 2, 0 | *CASCADE_SCALE_IMAGE*, Size(30, 30));

- Recadrage de l'image :

- Définition du région d'interêt :

  cv::Rect roi_b;

  cv::Rect roi_c;

- Boucler sur les visages détectés pour cadrer la plus grande zone possible ; L'algorithme peut détecter même des parties de visage, donc on doit boucler sur tous les éléments pour trouver l'image de dimensions maximal, cette image sera la plus grande partie du visage détecté.

- Sauvegarder le résultat (Si on est besoin de le sauvegarder)

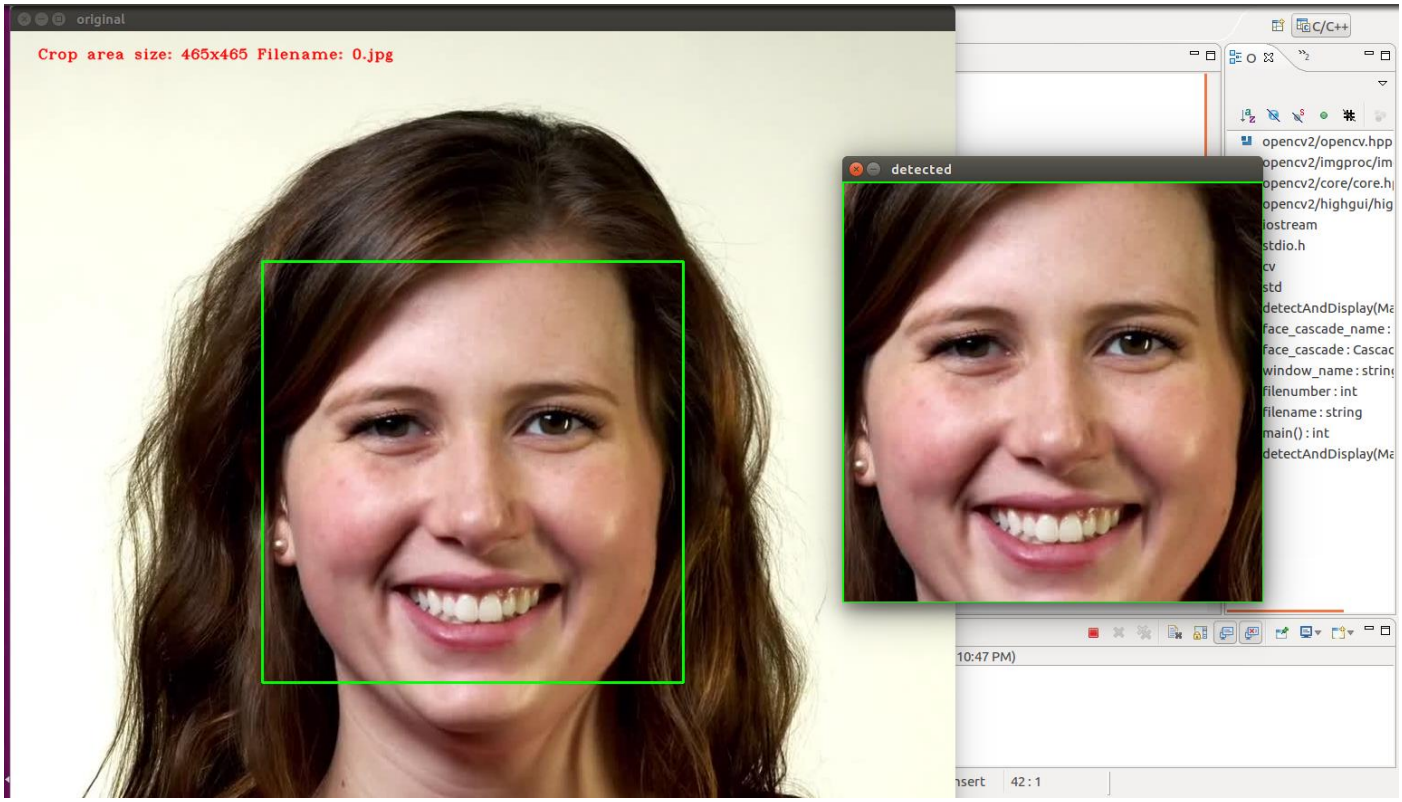


Figure 8: Résultat de l'exécution du code de détection du visage en utilisant la méthode Haar cascade

La détection du visage est la 1ère phase important dans la reconnaissance de l'expression faciale. La prochaine phase sera l'utilisation du visage détectée dans l'un des algorithmes de reconnaissance des émotions pour aboutir à notre résultat. Pour vérifier que la détection du visage marche bien et plus précisément détecte seulement un visage clair d'un être humain (car les algorithmes des reconnaissances des expressions faciales se basent sur le visage des humains dans leurs calcul pour

obtenir un résultat), on va la tester avec 2 images négatives : une qui ne contient pas de visage et l'autre contient un visage pris d'un dessin animé.

Figure 9: Image négative 1 : pas de visage

Résultat :

OpenCV(3.4.1-dev) Error: Assertion failed (found && "Can't destroy non-registered window") in cvDestroyWindow, file /home/saddiki/opencv-workspace/opencv/modules/highgui/src/window_gtk.cpp, line 1257

terminate called after throwing an instance of 'cv::Exception'

what(): OpenCV(3.4.1-dev) /home/saddiki/opencv-workspace/opencv/modules/highgui/src/window_gtk.cpp:1257: error: (-215) Assertion failed: found && "Can't destroy non-registered window" in function cvDestroyWindow

Figure 10: Image négative 2 : Dessin animé

Résultat :

```
OpenCV(3.4.1-dev) Error: Assertion failed (found && "Can't destroy non-registered window") in cvDestroyWindow, file /home/saddiki/opencv-workspace/opencv/modules/highgui/src/window_gtk.cpp, line 1257

terminate called after throwing an instance of 'cv::Exception'

  what():  OpenCV(3.4.1-dev) /home/saddiki/opencv-workspace/opencv/modules/highgui/src/window_gtk.cpp:1257: error: (-215) Assertion failed: found && "Can't destroy non-registered window" in function cvDestroyWindow
```

Détection du visage en utilisant Opencv et Python (Méthode Local Binary Patterns + Support Vector Machines) :

Aspect théorique

Cette méthode repose sur la computation locale des textures d'une image, et est bien plus rapide que la méthode précédente (Haar Cascade) bien que moins précise (10 à 20 % moins performant que Haar en général).

Voici les étapes pour extraire les LBP features d'une image et utiliser ces derniers pour faire de la reconnaissance de visages :

Etape 1 : Transformation de l'image couleur en image en niveaux de gris

Etape 2 : Pour chaque pixel de l'image, appliquer l'algorithme suivant :

Comparer la valeur du pixel avec ses voisins.

Remplacer les pixels ayant une plus grande valeur strictement que le pixel central par 0, les autres par 1, et le pixel central par null.

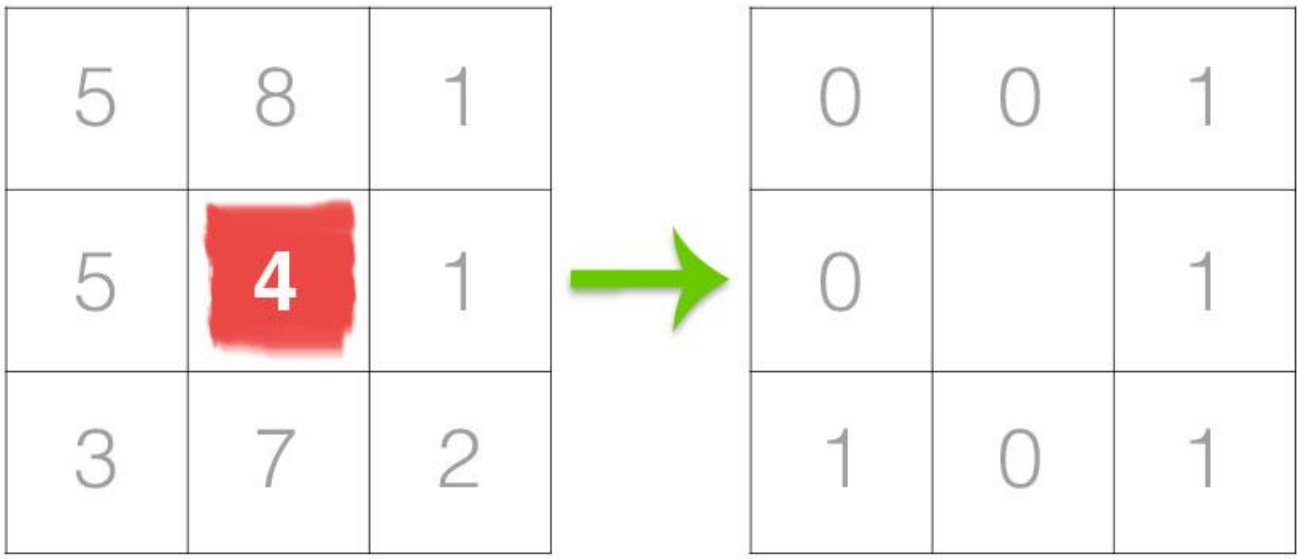


Figure 11: Application de l'algorithme

Stocker les pixels dans un tableau de 8 cases, et convertir la chaine de 0 et 1 en décimal.

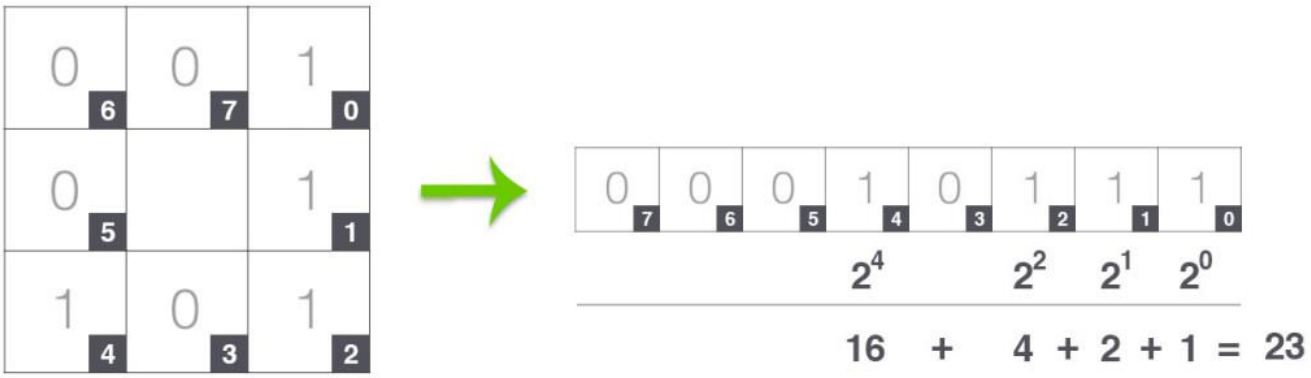


Figure 12: Calcul de la nouvelle valeur

Remplacer la valeur du pixel central par la valeur obtenue.

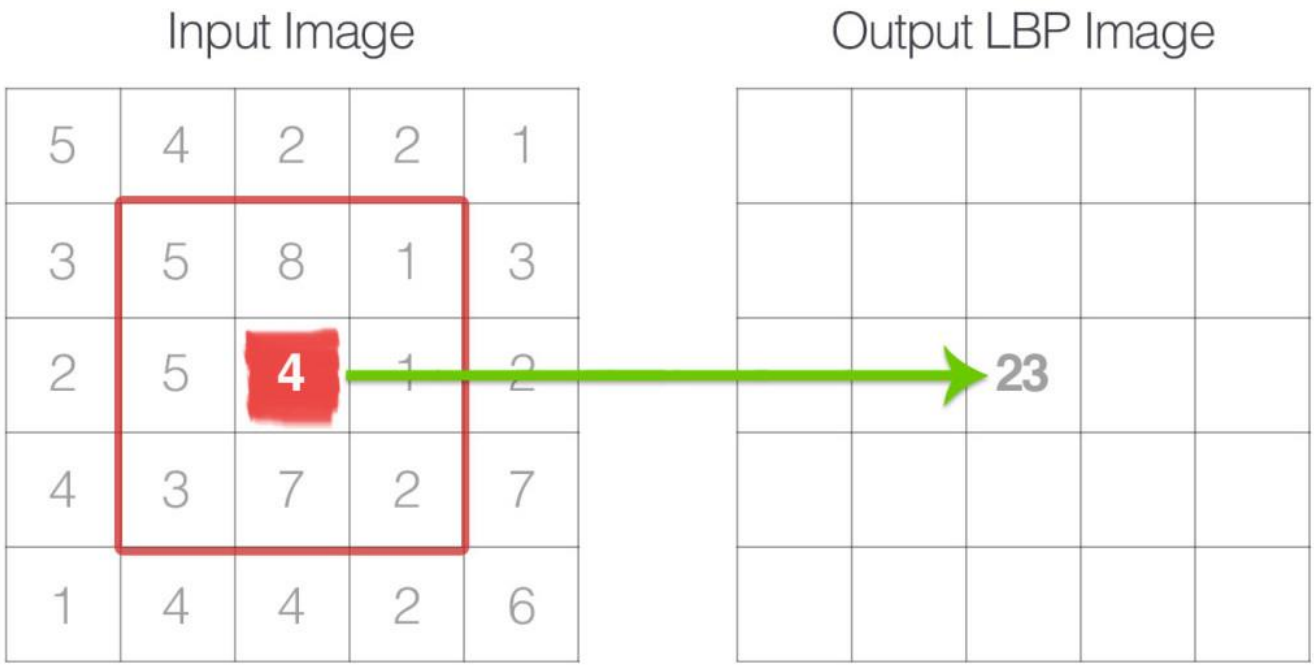


Figure 13: Remplacement de la valeur initiale du pixel

L’image résultante ressemble à ceci :

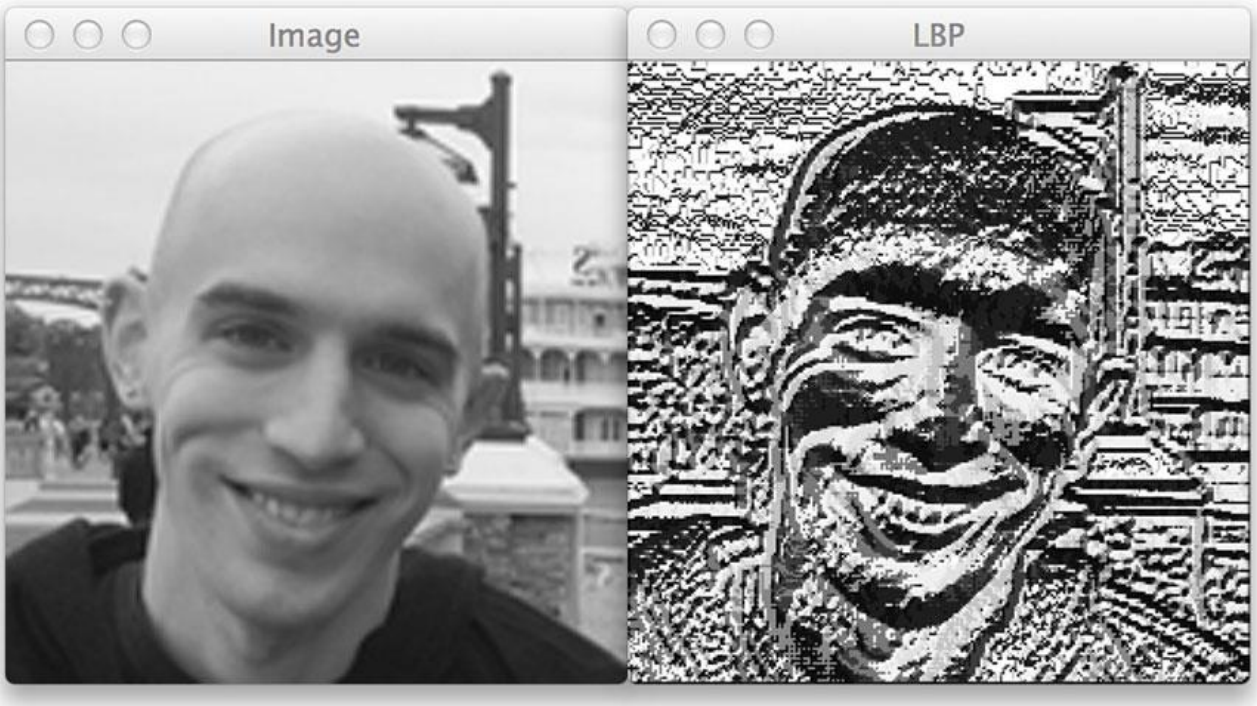


Figure 14: Image avant et après extraction des LBP features

Etape 3 : Convertir l'image résultante en histogramme

Etape 4 : Utiliser Support Vector Machines en tant que « classifier »

Dans le monde du Machine learning, le SVG est fréquemment utilisé pour regrouper des objets de même type et séparer les objets de types différents, en utilisant un set d'objets déjà classifié (training set) pour prédire la classification d'un set non classifié (testing set).

Etape 5 : Extraire les LBP features de plusieurs visages et passer leurs histogrammes à l'algorithme SVG pour l'entrainer

Ces visages représentent le « training set » permettant de préparer le SVG a la classification de nouvel input, le « testing set ».

Détection du visage basé sur la couleur de la peau

Aspect théorique

Cette méthode est très rarement utilisée car l'algorithme détectant le visage n'est pas assez précis, est sujet à des erreurs, et a besoin comme input d'une image contenant déjà un visage occupant une bonne

partie de l'écran et est de bonne qualité, alors que les méthodes précédentes permettent de détecter la présence (ou l'absence) de plusieurs visages (flous ou pas) dans une seule image.

Néanmoins, son implémentation est très simpliste et très utile dans les petits projets où la détection précise de visage n'y représente pas une partie centrale.

En effet, il suffit de rentrer une image contenant au plus un visage en input et appliquer un filtre pour séparer la partie visage du reste, voici un exemple :

- **RGB Model:** 0.836G – 14 < B < 0.836G + 44 => Skin OR 0.79G -67 < B < 0.78G + 42 => Skin
- **HIS/HSV Model:** 19 < H < 240 => Not skin
- **YCrCb Model:** 140 < Cr < 165 => Skin OR 140 < Cr < 195 => Skin

Ce qui donne le résultat suivant :

Fig 3.Orignal Image

Fig.4 Image after Skin Thresholding

Figure 15: Application du filtre a une image

L'output est une image en noir et blanc. On peut constater que certaines zones dans l'image sont faussement attribuées à un visage, chose qui peut être réglée en appliquant un algorithme d'érosion a l'image, mais fait perdre en qualité a celle-ci.

Détection du visage en utilisant la méthode HOG (Histogram of Oriented Gradients)

Cette méthode, inventée en 2005, fait partie des méthodes « modernes » de reconnaissance de visages, et est utilisée de nos jours par Facebook.

Aspect théorique

Etape 1 : Convertir l'image input en nuances de gris

Figure 16: Image source convertie en nuances de gris

Etape 2 : Remplacer chaque pixel de l'image par une flèche (gradient) indiquant la direction par laquelle l'image devient plus sombre

Consiste à appliquer à chaque pixel un filtre dérivatif 1-D centré, dans les directions horizontales et verticales. Les masques suivants sont utilisés pour cela :

$[-1, 0, 1]$ et $[-1, 0, 1]^T$

Figure 17: Œil droit avant calcul des gradients

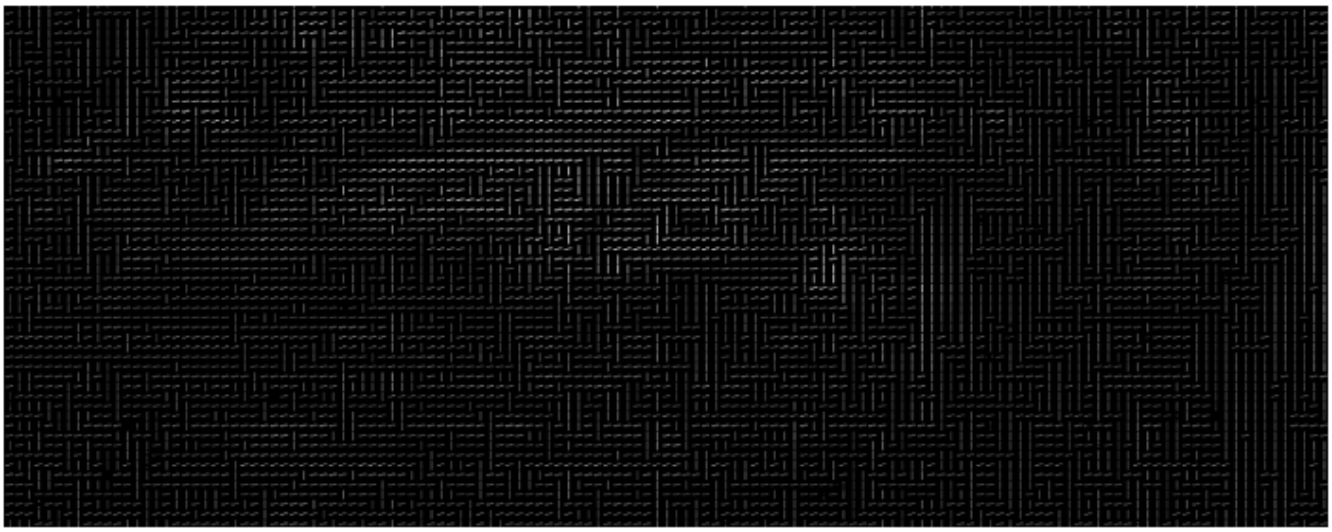


Figure 18: Œil droit après calcul des gradients

Ce calcul, appliqué à chaque pixel, représente un trop grand nombre d'opérations et est bien trop précis, impactant négativement les performances de l'algorithme, c'est pourquoi on préfère décomposer l'image en portions 16 x 16 chacune et calculer la direction majeure vers laquelle cette portion devient plus sombre, au lieu de faire ce calcul pour chaque pixel.

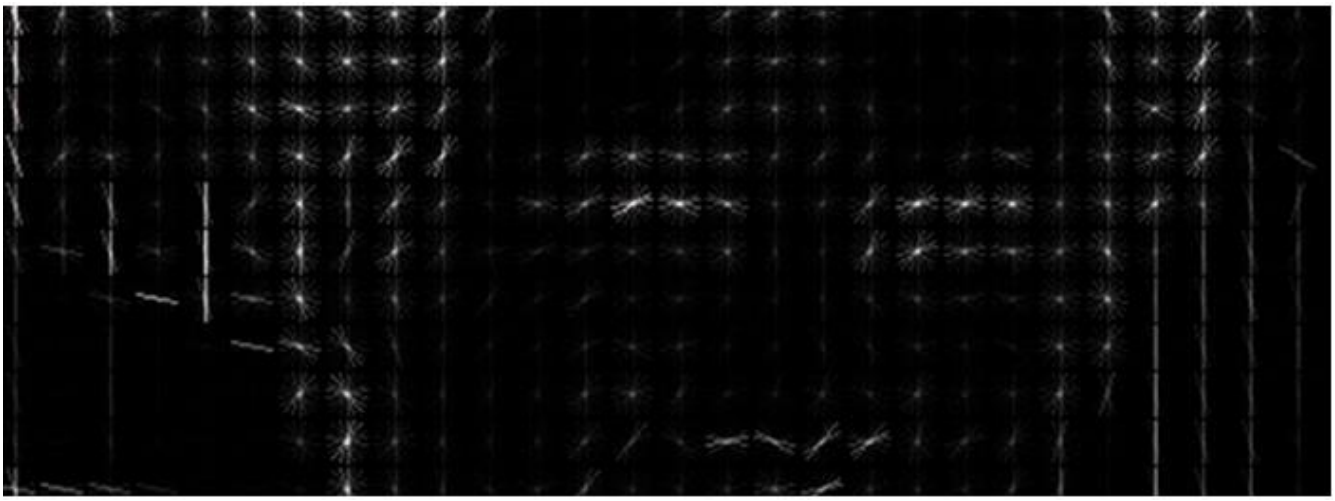


Figure 19: Calcul des gradients pour chaque portion de 16 x 16 de l'image

Etape 3 : Comparer l'image en output avec d'autres représentation HOG d'images contenant au moins un visage

Si la représentation HOG d'une image passée en input a un algorithme de comparaison avec d'autres représentations HOG d'un visage dans la base de données, alors on conclut que l'image contient un visage.

Conclusion du chapitre

Ces différentes méthodes de reconnaissance de visages, nombreuses et variées en termes d'ingéniosité et d'algorithmes et procédés utilisés sont la base a partir de laquelle se basera la détection d'expressions faciales dans le chapitre suivant.

Chapitre II

Reconnaissance des expressions faciales

II- Reconnaissance des expressions faciales :

1- Algorithmes :

1-1 IBM Watson Visual Recognition service :

IBM Watson Visual recognition service est un service cloud d'IBM qui permet la détection et la reconnaissance des objets, des visages et même déterminer ou le sexe ou bien donner une estimation sur l'âge d'une personne.

Ce service peut être utile dans plusieurs domaines comme :

→ L'industrie : Vérifier que les produits sont bien conformes aux normes de fabrication.

→ Sécurité : Utiliser le service dans la détection des véhicules ou bien des criminels en utilisant les caméras de surveillance

→ Social Commerce : Utiliser l'image d'un plat pour trouver le restaurant où il a été servi ou bien utiliser la photo d'une vacance pour trouver des endroits similaires

La détection des objets en utilisant IBM Watson est possible selon 2 cas : L'objet existe déjà dans la base de données de la méthode ou bien l'objet n'existe pas dans la base de données de la méthode et dans ce cas on doit utiliser un ensemble d'images positives et négatives comme suit :

Cas 1 : Les attributs/objets existent déjà dans la base de données :

Détection d'objets :

En utilisant un simple code java on peut facilement détecter les attributs déjà définis dans la base de données standard dans une image :

```
VisualRecognition service = new VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);
service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/api");
service.setApiKey("your_api_key_here");
```

// Dans cette partie de code on utilise la clé fournit par IBM (en s'inscrivant à IBM Bluemix) qui donne à notre application l'accès au service Visual Recognition

```
String imageURL = new
String("https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/fruitbowl.jpg");
ClassifyImagesOptions options = new ClassifyImagesOptions.Builder().url(imageURL).build();
VisualClassification result = service.classify(options).execute();
```

// Importer l'image à utiliser dans le code et passer cette image comme paramètre au Classifier

Figure 20: Image source

```
System.out.println("Classification Results:");
System.out.println(result);
```

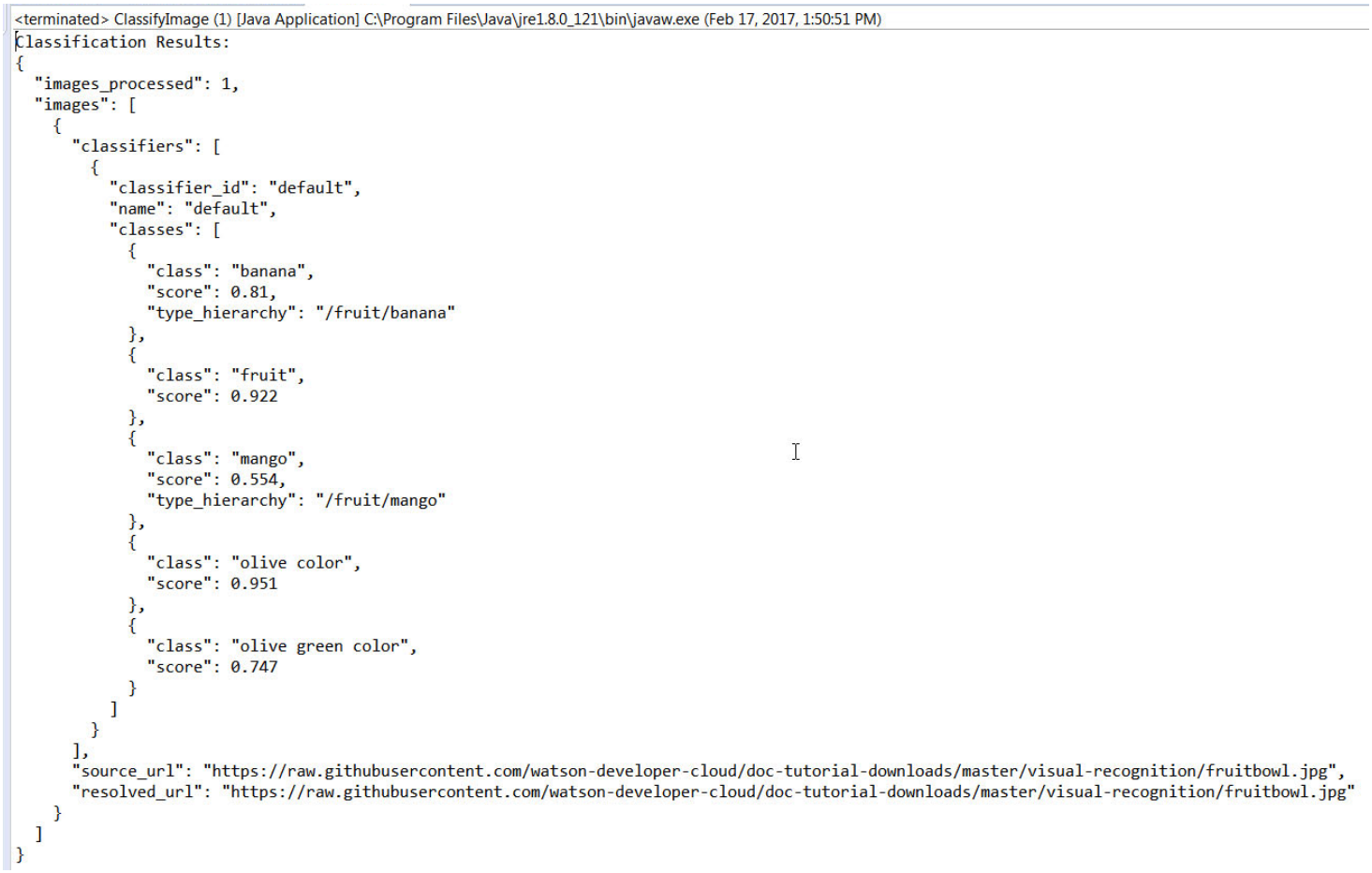


Figure 21: Résultat

Détection et reconnaissance d'un visage :

La détection des visages en utilisant IBM Watson (en utilisant Eclipse IDE pour Java) passe par 4 étapes :

- Passer l'image en argument à partir d'un lien.
- Définir la clé de l'API de Visual Recognition Service.
- Passer l'image à la méthode *detectFaces* de Visual Recognition Service pour qu'elle sera traité.
- Retourner le résultat en format JSON.

Figure 22: Image source

Figure 23 : Résultat de détection des visages

Les attributs « face_location », « age », « gender » et « identity » sont définis par défaut dans la base de données, et le « classifier » est entrainé à reconnaitre ces attributs-là, mais on peut rajouter nos propres attributs, comme les expressions faciales des individus présents dans l'image. En effet, les algorithmes de IBM Watson Visual Recognition Service sont désormais assez puissants pour détecter des émotions dans l'image. Il suffit d'entrainer le classifier pour reconnaitre ces derniers.

Le cas 2 détaille la façon d'ajouter un attribut a la base de données.

Cas 2 : L'attribut/objet n'existe pas dans la BD :

Dans ce cas en doit entrainer le Classifier c'est à dire ajouter le nouvel objet à sa base de données, cela se fait en utilisant deux dossier contenant plusieurs images, un dossier pour les images positives et l'autre pour les images négatives, en utilisant ces ensembles des images la méthode peut trouver les similarités entre les images pour définir l'objet.

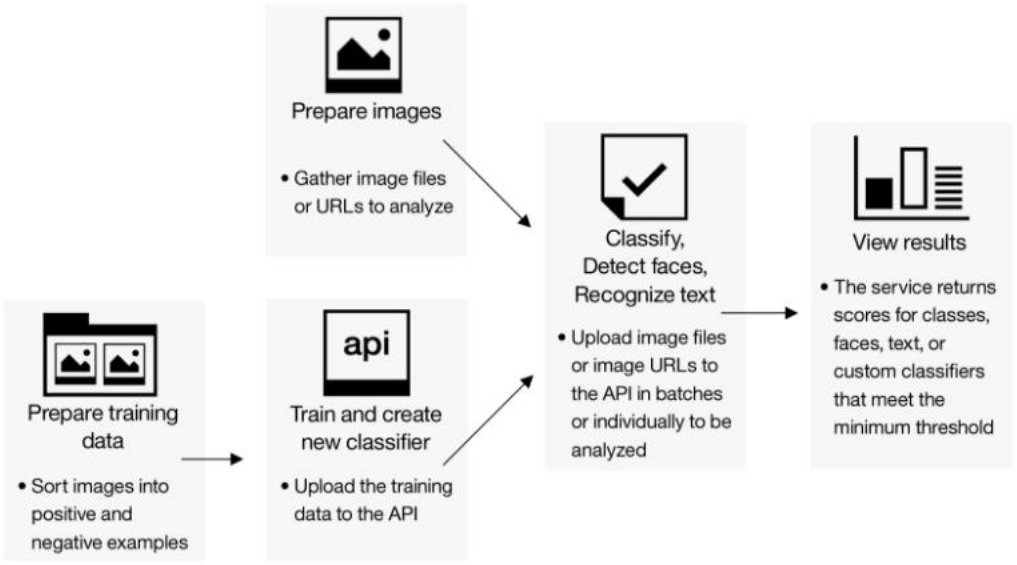


Figure 24: Schéma détaillant le processus de reconnaissance d'objets d'IBM Watson

En utilisant cette méthode, on peut entraîner le Classifier en utilisant un ensemble d'images contenant des différentes expressions faciales pour qu'ils soient utilisés après dans la reconnaissance des émotions, mais les images doivent respecter les différences entre les êtres humains, à savoir :

- Le sexe
- La couleur de la peau
- La couleur des cheveux
- La couleur des yeux
- Barbe/Moustache

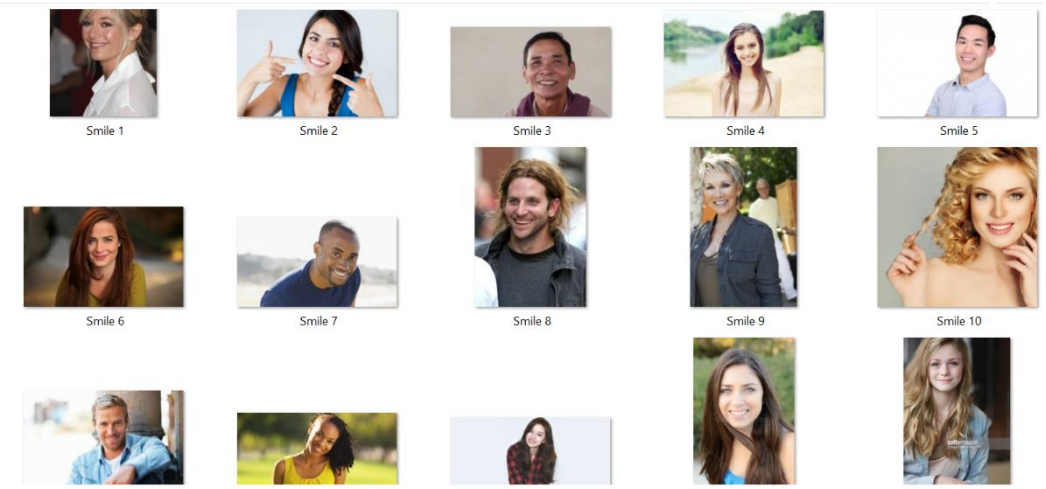


Figure 25: Ensemble d'images de personnes affichant une expression joyeuse

2-2 Détection de l'émotion à partir de la forme de la bouche :

Cet algorithme se démarque par sa simplicité et son efficacité malgré l'input limité sur lequel il travaille, en effet, comme indiqué dans le titre, cette méthode de détection n'utilise que la forme de la bouche pour reconnaitre une émotion, et a besoin d'une image ne contenant qu'un seul visage au plus, car l'algorithme ne traite qu'un seul visage a là fois.

Voici les étapes que suit cette méthode :

Etape 1 : Détection du visage

Voir Chapitre 1, partie « Détection du visage basé sur la couleur de la peau »

Etape 2 : Appliquer l'algorithme de dilatation et d'érosion a l'image en output pour éliminer le bruit et les zones ne faisant pas partie du visage

Fig.5 Image after Dilation

Fig.6 Image after Erosion

Figure 26: Application de l'érosion et la dilatation a l'image

Etape 3 : Rogner l'image au niveau du visage

Permet d'éliminer toute trace de couleur rouge dans l'arrière-plan, pour éviter l'interférence de de dernier dans les étapes suivantes.

Etape 4 : Détecter les lèvres de l'individu

La méthode utilisée est la même que celle utilisée pour détecter le visage de l'individu, c.-à-d. l'utilisation de filtres de couleur rouge pour détecter la zone ou les lèvres se situent.

Cette étape est la raison pourquoi il est préférable d'utiliser cet algorithme sur des images ne contenant au plus qu'un seul individu.

Etape 5 : Rogner la zone des lèvres

Figure 27: Zone des lèvres rognée

Etape 6 : Comparer l'image résultante a une base de données contenant déjà des échantillons d'images de lèvres associées à une émotion en particulier

L'émotion contenant l'échantillon ayant le plus de similarité avec l'input est considéré comme l'émotion associée à ce dernier.

Algorithme de détection d'émotions en utilisant la logique floue

L'algorithme de reconnaissance d'expressions faciales passe par 5 étapes :

Etape 1 : Appliquer un filtre de couleur floue et extraire la zone faciale en utilisant la méthode d'analyse des histogrammes

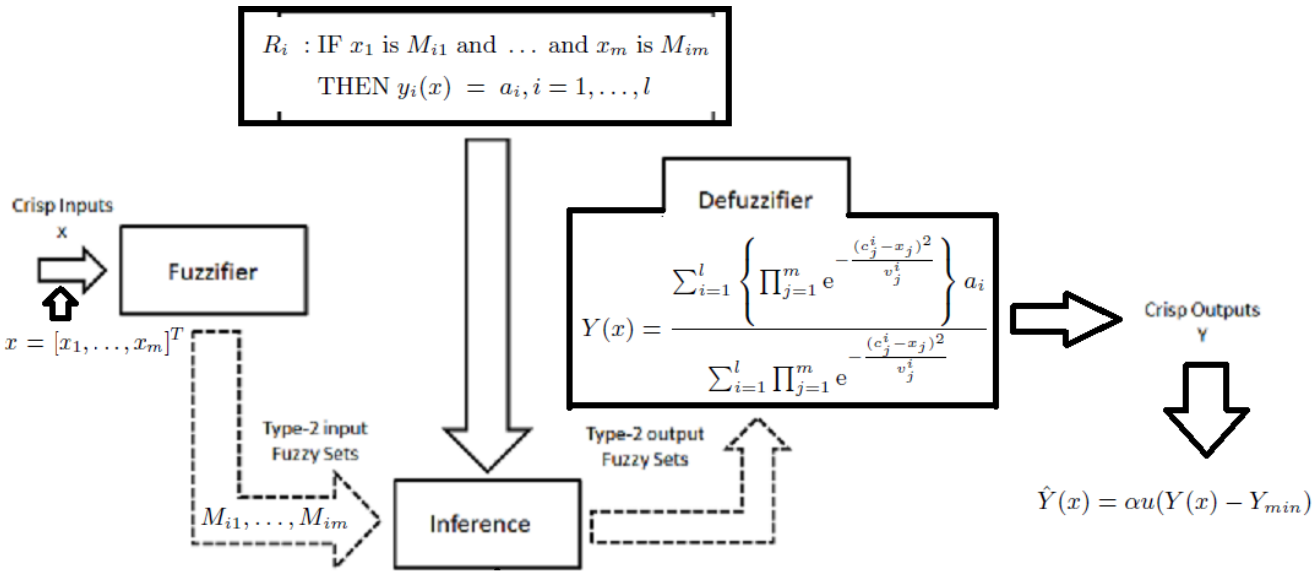
Afin de détecter un ou plusieurs visages dans une image passée en input, l'algorithme applique un filtre afin de détecter la zone correspondante a la couleur de peau des individus et ainsi rogner la région faciale.

Le filtre est basé sur la théorie des « fuzzy inference systems » qui, dans le monde du machine learning, permet d'assigner à un objet des caractéristiques de façon probabiliste et non stricte.

Par exemple, dans un modèle strict, seule la valeur true et false peut être appliqué à un objet, et rien au milieu, ce qui n'est pas applicable au monde réel. Mais dans un modèle flou (fuzzy), Un objet peut être 0.7 true et 0.3 false.

Le filtre pourra donc recevoir en input une série de couleurs correspondant aux couleurs détectées sur l'image, et ainsi, en passant par un algorithme flou, pourra détecter les zones ou la couleur se rapproche le plus à un type de couleur de peau connu (0.8 blanc et 0.2 caucasien par exemple).

L'algorithme suit ces étapes :



where $x_i \in \mathbb{R}$ is the ith color input, $M_{i1}, \ldots, M_{im}$ are the antecedent fuzzy sets, $y_i(x)$ is the consequent output of the ith rule, $x = [x_1, \ldots, x_m]^T \in F \subset \mathbb{R}^{\geqslant}$ is the input feature vector, F is the feature vector set, and a_i is the consequent parameter and mean the weight for rule i.

where c_j^i and v_j^i are the center the width of membership function of the jth feature in the ith rule.

where α is the offset value for gray image, $u(x)$ is the unit step function, and Y_{min} is minimum value of $Y(x)$. Therefore, if $Y(x)$ is greater than Y_{min}, the final output has α. Adjusting Y_{min}, we can change robustness of skin color filter.

Après application du filtre, l'image résultante est convertie en un histogramme en utilisant les niveaux de gris de l'image, puis, en choisissant le plus grand segment de l'histogramme selon un algorithme, on peut cerner la zone faciale avec une précision optimale.

Etape 2 : Extraire les composants du visage en utilisant la méthode VFM et la méthode d'analyse des histogrammes

La méthode VFM (Visual Face Model) consiste à transformer un visage 2D en un visage 3D, offrant plus de possibilités pour analyser le modèle en output. A partir de ce modèle, il devient facile d'extraire des zones précises du visage (nez, yeux, bouche…)

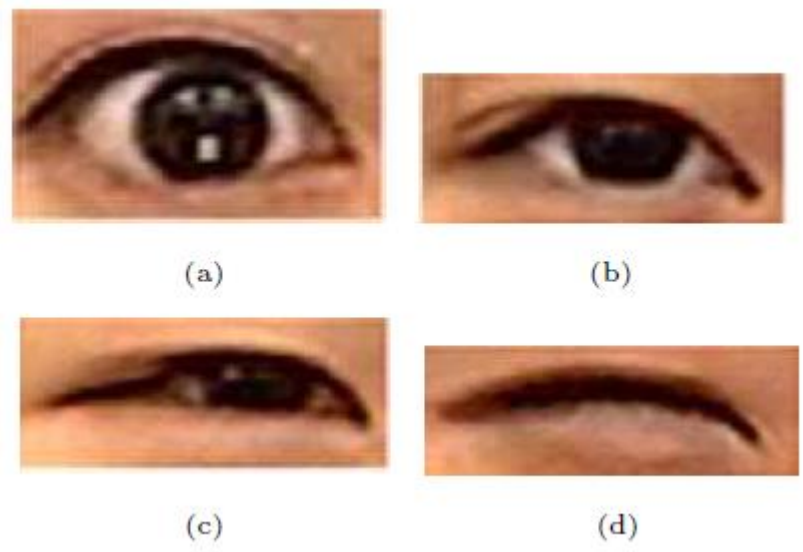
Etape 3 : Extraire les vecteurs et les caractéristiques des composants du visage détecté

Les caractéristiques (features) extraites sont divisées en trois catégories :

Caractéristiques de la zone des yeux

Features	Description	Size
X_{e1}	Distance between two eye brow	1×1
X_{e2}	Distance between eye and eye brow	1×1
X_{e3}	Distance between nose and eye(left side)	1×1
X_{e4}	Distance between nose and eye(right side)	1×1
X_{se}	Error between eye and template	4×1

Des templates sont utilisés pour connaitre la forme des yeux, leur taille et leur état (fermé, ouvert, semi-ouvert…)



La comparaison de la zone avec les templates se fait en utilisant une fonction de similitude S

$$S = |X_w - T_w| + |X_h - T_h| + \left| \frac{X_w}{X_h} - \frac{T_w}{T_h} \right| + |X_p - T_p|$$

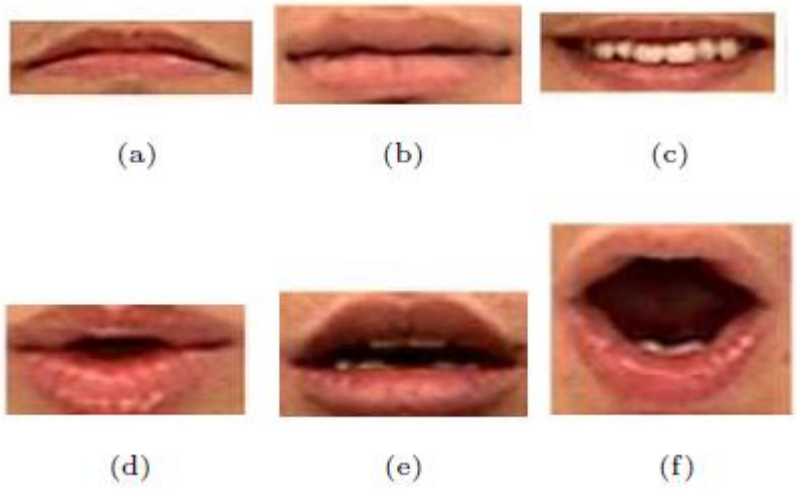
Let X_w, X_h, and X_p are width, height, and the number of pixel in image.

where T_w, T_h, and T_p are width, height, and the number of pixel in template.

Caractéristiques de la zone buccale

Features	Description	Size
X_{m1}	$\frac{\text{Width of mouth}}{\text{Height of mouth}}$	1×1
X_{m2}	Distance between nose and mouth	1×1
X_{se}	Error between mouth and template	6×1

Idem, des templates sont utilisés pour connaitre la forme de la bouche, sa taille/longueur/largeur et son état



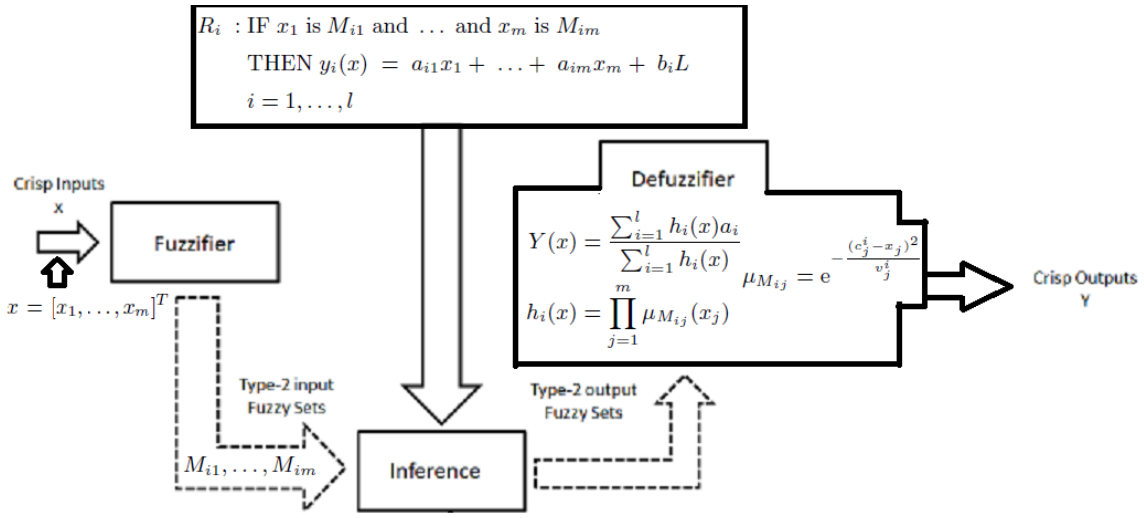
Caractéristiques des zones auxiliaires

Features	Description	Size
X_{a1}	Existence of winkles between eyes	1×1
X_{a2}	Existence of winkles in left cheek	1×1
X_{a3}	Existence of winkles in left cheek	1×1

Etape 4 : Identifier un « classifier » flou avec les vecteurs extraits

De la même façon que dans l'Etape 1, on utilise un algorithme basé sur les « fuzzy inference systems » pour associer une expression faciale a une des émotions stockées dans la base de données.

L'input est composé des features calculés dans l'étape précédente, et l'output est sous la forme de plusieurs valeurs associées à une émotions en particulier, l'émotion la plus présente est considérée comme l'émotion principale.



where $x_i \in \mathbb{R}$ is the ith color input, $M_{i1}, \ldots, M_{im}$ are the antecedent fuzzy sets, $y_i(x)$ is the consequent output of the ith rule, $x = [x_1, \ldots, x_m]^T \in F \subset$ is the input feature vector, F is the feature vector set, and a_{ij} and b_i is the consequent parameters.

where $h_i(x)$ is the firing strength of the ith rule and $\mu_{M_{ij}}(x_j)$ is the membership degree of the jth feature of the ith rule.

where c_j^i and v_j^i are the center the width of membership function of the jth feature in the ith rule.

Etape 5 : Tester le « classifier » flou

Après avoir défini un set pour entrainer le classifier, il ne reste plus qu'à faire entrer un « testing set » pour détecter les émotions présentes dedans.

Conclusion du chapitre

Ces différentes méthodes de reconnaissance d'expressions faciales, variées et nombreuses, produisent un résultat différent en fonction des algorithmes qui y sont appliqués. Après ces deux longs chapitres destinés à la description théorique de la détection de visages et d'émotions, passons a la pratique.

Chapitre III

Réalisation d’un programme de détection d’expressions faciales

III- Réalisation d'un programme de détection d'expressions faciales

Après avoir détaillé certaines méthodes de détection de visage et de détection d'expressions faciales, il est temps d'implémenter une de ces méthodes dans un programme sous Opencv en utilisant le langage C++.

Méthode utilisée pour la détection faciale

Après avoir pesé les pours et les contres de chaque méthode décrits précédemment dans le chapitre In nous avons décidé de nous reposer sur la méthode de détection faciale basée sur les Haar-Cascades, en effet, leur flexibilité leur permet d'être utilisées non seulement pour la détection de visages, mais aussi pour la détection des yeux et de la bouche (chose que les autres méthodes ne garantissent pas nécessairement) ce qui nous sera crucial pour l'étape suivante de détection d'émotions. De plus, leur utilisation est répandue et populaire, si bien que la bibliothèque Opencv possède des méthodes et des fichiers spécialisés pour ce type de méthode.

Méthode utilisée pour la détection d'expressions faciales

La détection d'expressions faciales est infiniment plus complexe que la détection de visage, nous ne pouvons pas nous rabattre sur de simples Haar-Cascades pour les décrire, c'est pourquoi nous avons décidé d'utiliser la méthode SIFT pour détecter les points d'intérêt d'une zone du visage, de calculer ses descripteurs et de comparer la distance de ces descripteurs avec une base de données d'échantillons de visages affichant une expression faciale particulière (joie, peur, tristesse…). La méthode SIFT est utilisée en particulier car les descripteurs calculés sont invariants à la rotation et a la taille de l'image, ce qui est très utile car une grande variété de tailles et d'orientations existe, même au sein d'un groupe de visages arborant la même expression faciale.

Outils utilisés

Système d'exploitation Linux (Ubuntu)

Utile de par ses nombreuses commandes, sa simplicité d'accès, sa rapidité et son performance.

Eclipse C/C++

Grace à son performance dans l'éxecution des codes, propose le nécessaire pour développer de nouveaux plug-ins, possibilité d'utiliser des outils open source et les outils d'aide au développement.

Bibliothèque Opencv

Bibliothèque de librairies (open source) incontournable spécialement conçue pour le traitement d'images et disponible pour plusieurs languages de programmation comme Python, C++, Java. Opencv est le choix naturel lors de la réalisation de ce projet vu l'ensemle des outils qui offre pour le traitement d'images et même des vidéos (Filtres prédéfinis, Decripteurs prédéfinis et plusieurs autre méthodes de traitement).

Réalisation

L'algorithme se déroule selon les étapes suivantes :

Etape 1 : Charger l'image

On utilise la méthode prédéfinit d'Opencv **imread(**`Chemin vers l'image`**)**

Etape 2 : Détection du visage

La détection du visage est expliquée en chapitre I.

Etape 3 : Détection des composants du visage (œil droit, œil gauche et bouche)

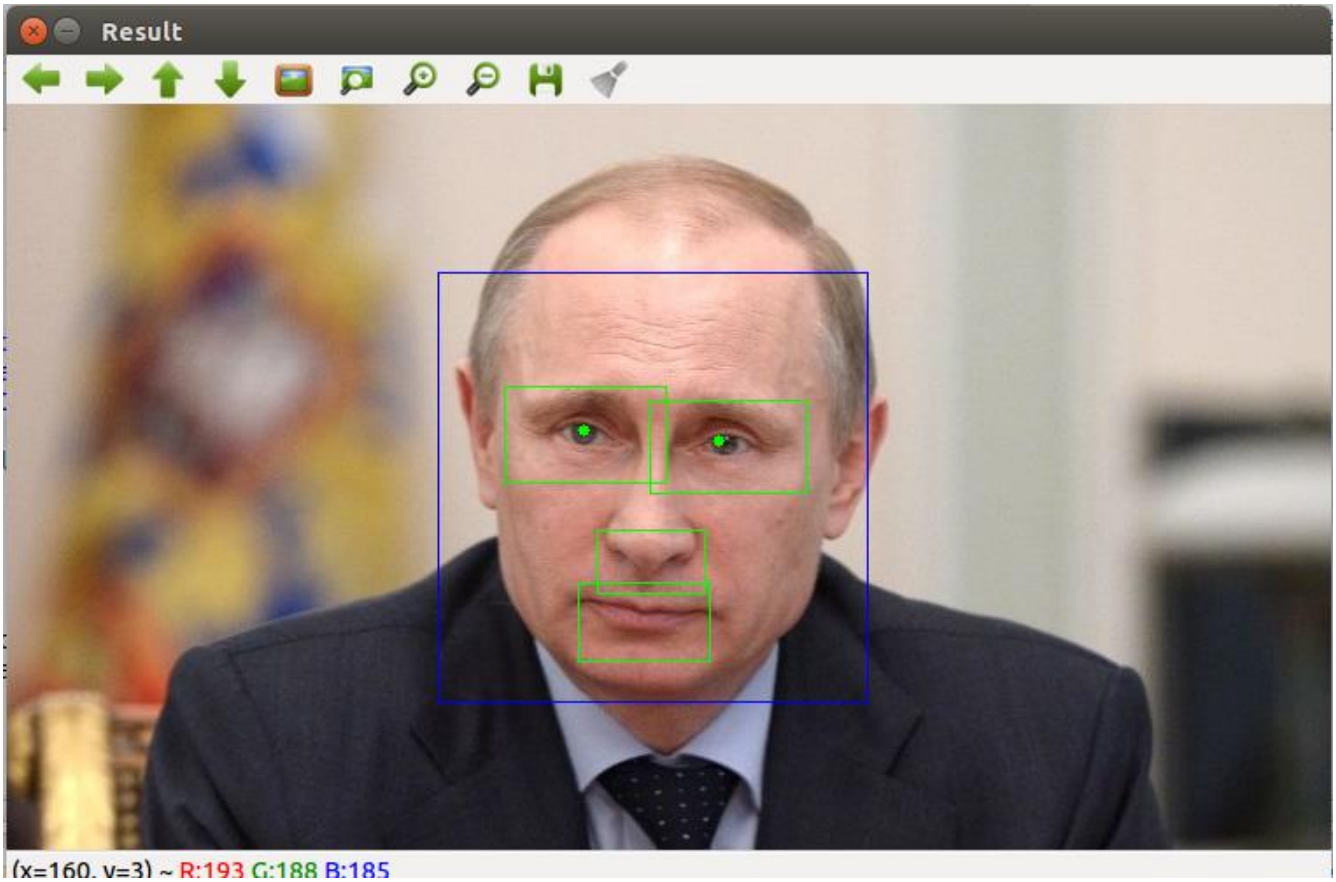


Figure 28: Image avec visage, yeux, nez et bouche detectés

Etape 4 : Conversion de l'image en niveaux de gris, puis rogner la zone de l'œil droit, de l'œil gauche et de la bouche séparément et sauvegarde en tant qu'échantillons sur lesquelles la détection d'expression faciale se fera

Etape 5 : Détection des points d'intérêt et calcul des descripteurs des échantillons selon la méthode SIFT

La librairie Opencv offre des méthodes prédéfinit pour le calcul des descripteurs selon SIFT

```
cv::SiftFeatureDetector detector;
vector<cv::KeyPoint> inputKeypoints;
detector.detect(img, inputKeypoints);
Mat inputDescriptors;
detector.compute(img, inputKeypoints, inputDescriptors);
```

Etape 6 : Comparaison de distance avec les échantillons d'images d'une base de données classée par émotion ayant subi les mêmes étapes précédentes

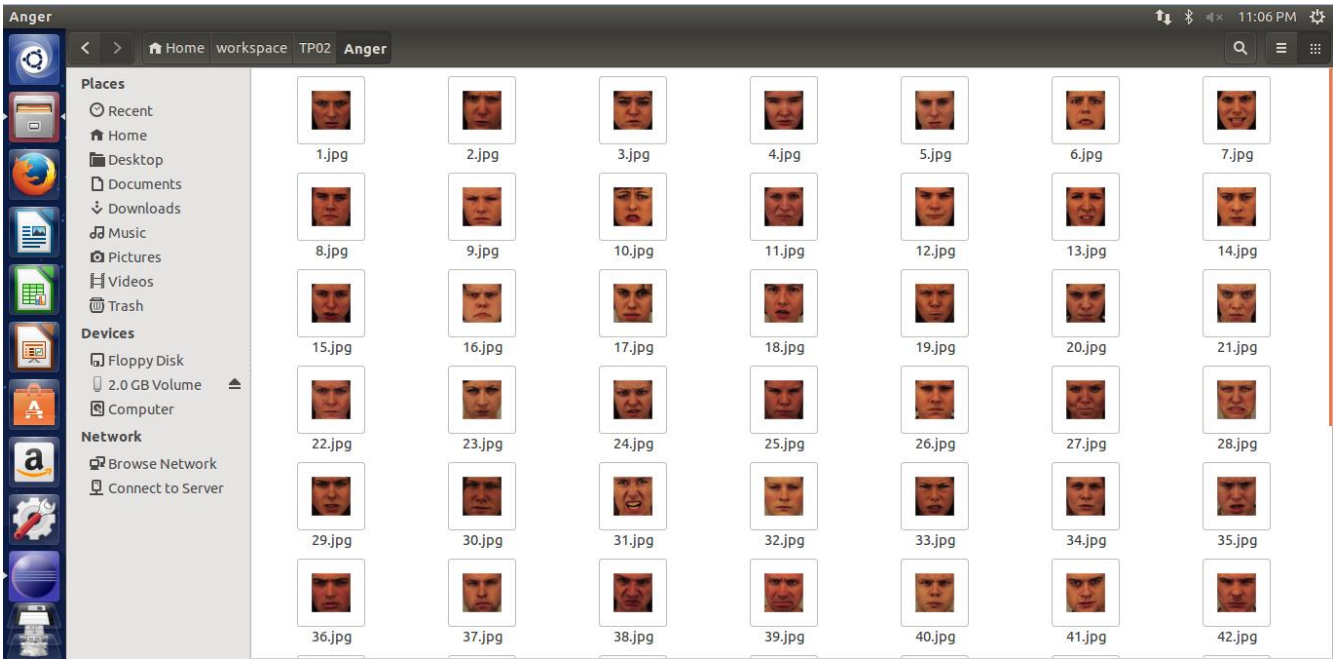


Figure 29: Base de données d'échantillons de personnes en colère

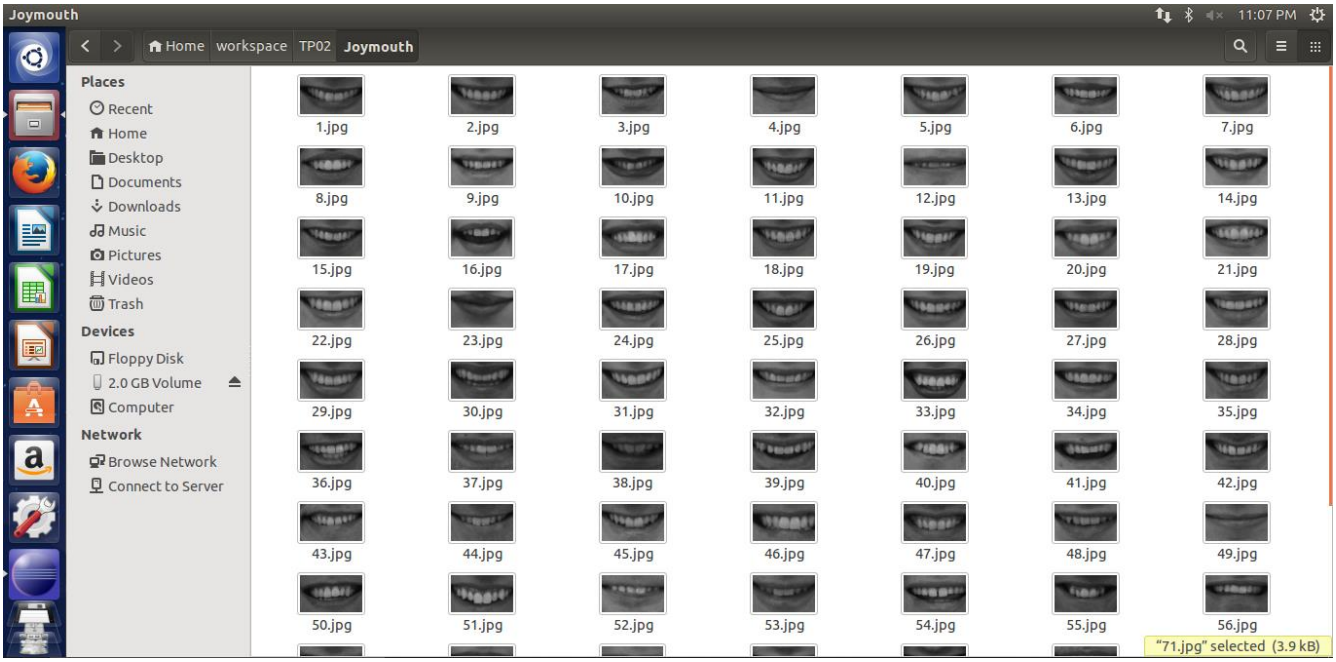


Figure 30: échantillons de bouches heureuses résultant de l'algorithme

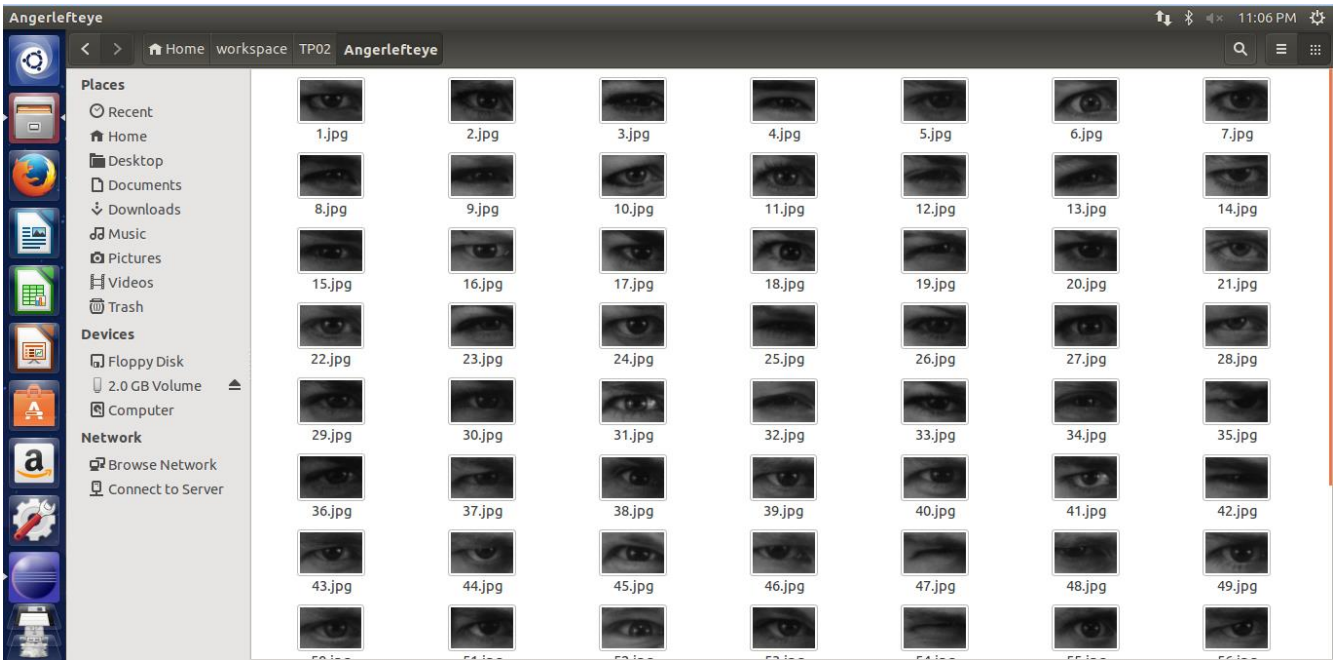


Figure 31: Yeux de personnes en colère

Etape 7 : L'échantillon le plus proche d'une émotion donnée est défini comme étant l'émotion du visage en input selon un pourcentage de ressemblance avec les images de la base

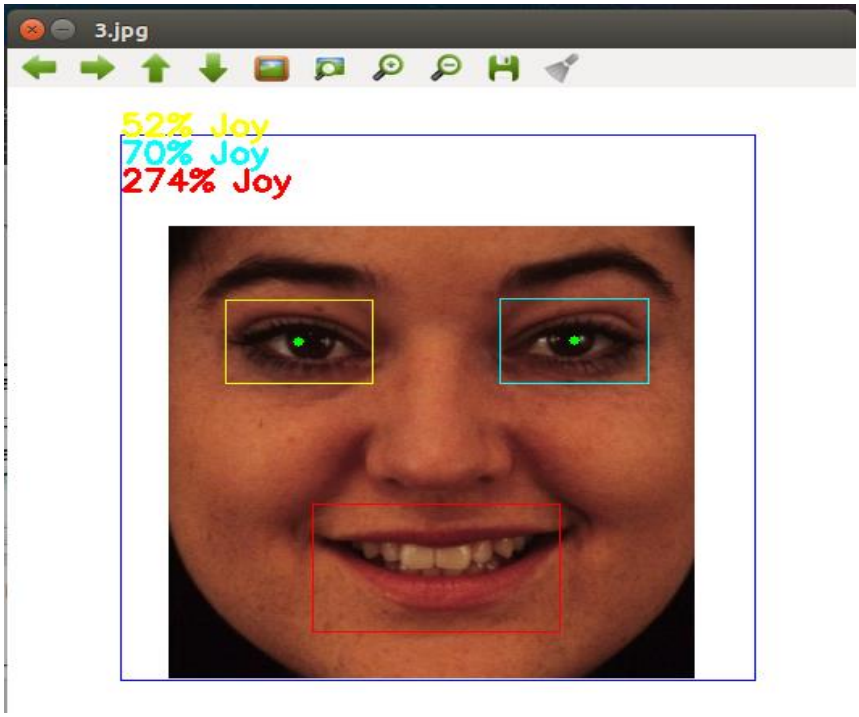


Figure 32: Image en output

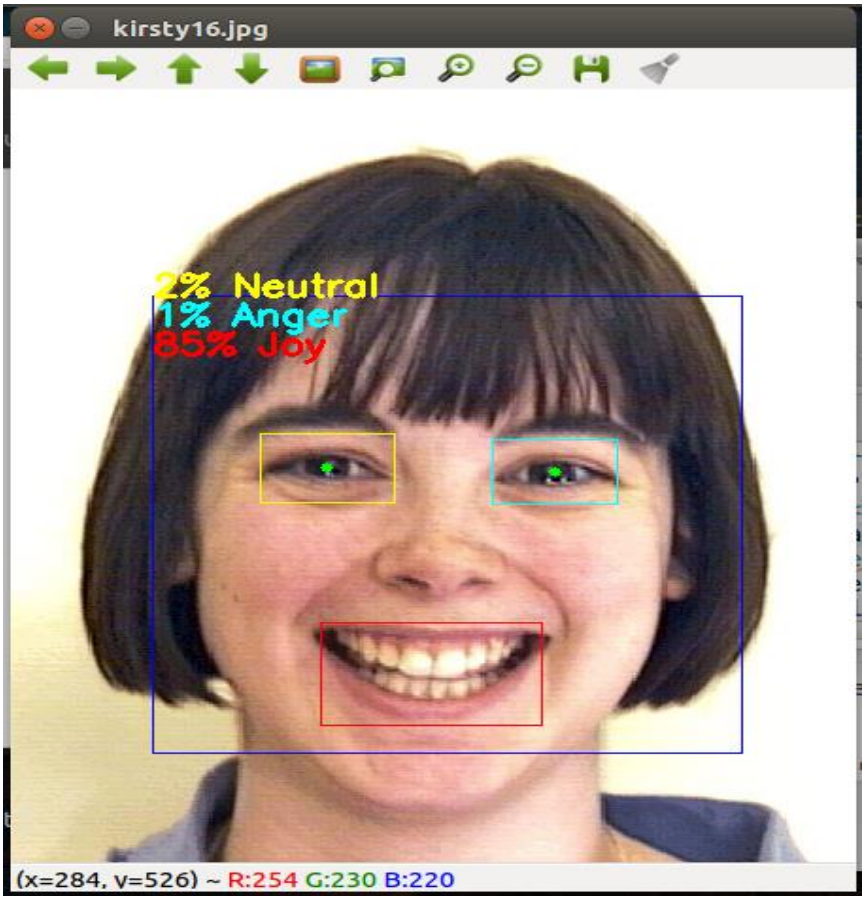


Figure 33: Image quelconque dont l'algorithme a été appliqué

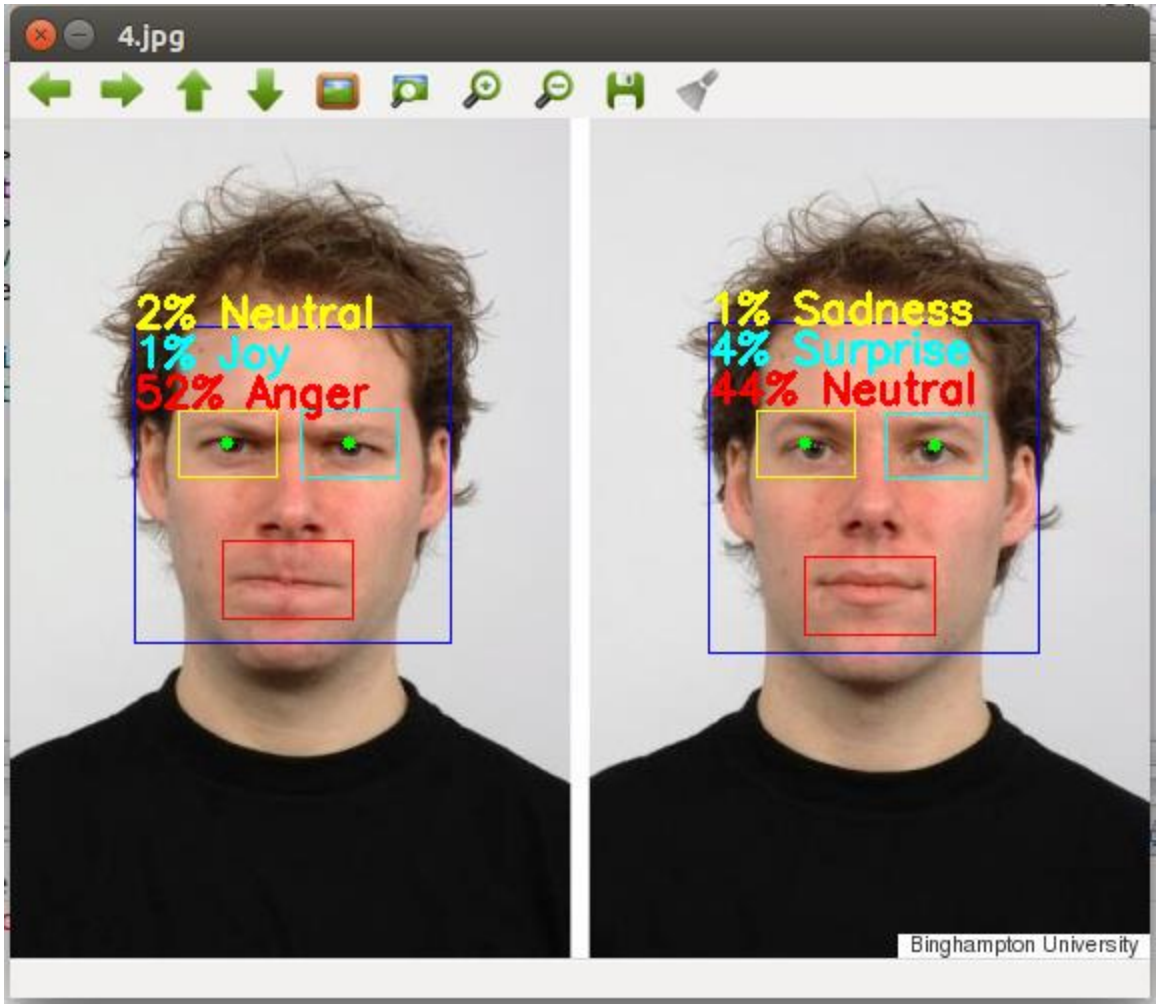


Figure 34: Image contenant deux visages sur laquelle l'algorithme a été appliqué

Conclusion du chapitre

L'algorithme décrit durant ce chapitre prend en input une image quelconque, applique les étapes décrites précédemment. La précision de l'algorithme dépend du nombre d'échantillons utilisés pour mesurer la distance entre l'image source et les images de la base de données, qui a leur tour rallongent la durée de traitement de l'image source, c'est pourquoi il faut trouver un compromis entre précision et performances en choisissant de bons échantillons.

Conclusion générale et perspectives :

En substance, ce projet de fin d'année consiste à développer une application de reconnaissance des expressions faciales. Nous vous avons présenté tout au long de ce rapport la démarche que nous avons suivie pour mettre au point cette application.

Toute fois pour améliorer notre application, on peut ajouter la possibilité de détecter et reconnaître l'expression faciale en temps réel en utilisant la webcam. On peut aussi ajouter une interface graphique GUI pour que l'utilisateur puisse choisir directement l'image à traiter depuis ses fichiers ou bien utiliser le " Drag And Drop " et non pas écrire le chemin de cette image à l'intérieur du code.

En outre, la réalisation de ce projet nous a été une opportunité de découvrir de nouveaux outils de développement, d'acquérir de nouvelles connaissances informatiques vu que c'est la première expérience avec un sujet qui concerne le traitement d'images.

Bibliographie :

- Cours Vision et perception numérique (Mme FKIHI)
- Cours Indexation des données multimédias (M ELHASSOUNY)
- www.stackoverflow.com
- www.researchgate.net
- Documentation Opencv
- IEEE Digital Library
- Livre: Building Cognitive Applications with IBM Watson Services: Volume 3 Visual Recognition