# ISEP - Projet C++
# Reconnaissance de Membres Humains

BERGER Thibault - SIZUN Pierre - TARLET Emilie - THAMBU Prathipan

15 décembre 2015

**Abstract**

L'objectif de ce module informatique a pour but de nous familiariser avec le langage C++, et notamment par l'utilisation de la bibliothèque OpenCV, axée sur le traitement d'image. Lors de ce projet, une application de reconnaissance de membres humain a été développée sous l'environnement de développement Eclipse.

Versions d'Eclipse: Juno (3.8), Mars (5.1)
Version d'OpenCV: 3.0

# Contents


I Cahier des charges . . . . . 2
II Présentation de l'algorithme . . . . . 2
II.1 Seuillage . . . . . 2
II.2 Transformations morphologiques et extraction de contours . . . . . 2
II.3 Reconnaissance des imagettes . . . . . 3
III Fonctions utilisées . . . . . 4
IV Conclusion . . . . . 5

# I Cahier des charges

L'application doit pouvoir reconnaître une main sur n'importe quel fond. Pour cela, il est nécessaire :

- D'utiliser un traitement adapté à la main afin de détecter au mieux ses contours
- De concevoir un algorithme qui permettra d'estimer si il y a bien une main sur une image
- D'extraire la main de l'image afin de ne conserver que celle-ci

# II Présentation de l'algorithme

## II.1 Seuillage

La peau humaine étant composée de pigments de couleur caractéristiques: beiges, marrons, noir, nous sommes partis sur l'idée d'une segmentation de la peau par pigmentation. Au cours de recherches bibliographiques, nous avons constaté qu'il existe une linéarité entre les différents pigments de peau existants sur chacun des canaux du domaine RGB.

En reprenant l'exemple donné sur le tutoriel d'OpenCV, un histogramme de couleur a pu être établi, et afficher les différents clusters dont nous aurions besoin pour chaque pigment. Cependant, en fonction de l'intensité lumineuse du fond (dépendant principalement de la présence de textures ou non), certains clusters se chevauchaient avec le bruit créé par la mauvaise capture de lumière. Lors de nos projets d'A2 basés sur l'OCR (Emilie, Pierre, Prathipan) ou la reconnaissance automatique de panneaux routiers (Thibault), nous avions déjà rencontré ce type de problème, et avons décidé de nous tourner vers un domaine de couleur plus robuste à ces variations: il s'agit du domaine YCbCr.
De même que pour le RGB, le domaine YCbCr se compose de 3 canaux:

- Y, basé sur la luminance, apportant des données de variance de luminosité
- Cb et Cr, deux composantes de chrominance, fournissant des données purement colorimétriques

Les pigments perçus par l'oeil humain dépendant principalement de l'intensité lumineuse alentours, le retrait de cette dernière laisse donc l'information utile aux données de chrominance, réduisant ainsi le nombre de clusters à 1.
En se basant donc sur ces deux données, et par calculs de ratio, nous avons déterminé un seuil, pour lequel les valeurs utiles sont conservées entre 0.6 et 0.9. Ces valeurs de pixels sont mises à 255, les autres à 0. Enfin, l'image originale modifiée est convertie en image binaire afin d'effectuer des traitements morphologiques.

## II.2 Transformations morphologiques et extraction de contours

Une fois le seuillage déterminé et appliqué, il faut extraire les éléments détectés afin de les analyser et décider de leur rapprochement à la main. Cependant, à ce stade, malgré la robustesse du domaine YCbCr, la probabilité de présence de pixels parasites, donc de mauvaises détections, reste élevée. Pour atténuer cette probabilité, il est nécessaire d'appliquer des opérations morphologiques (érosion, dilatation, ouverture, fermeture).
Cette étape de tests effectuée, il reste à obtenir les contours fermés intéressants. La détection de contours peut être appréhendée de plusieurs manières:

- Une approche gradient avec le calcul du gradient vertical pour horizontal avec les noyaux de Prewitt ou Sobel (rajout d'un lissage précédant le calcul du gradient)
- Une approche par filtre Laplacien.

Le choix s'est tourné vers l'opérateur Laplacien pour plusieurs raisons:
- Dans le cas d'une détection de contours sur fond uniforme, le seuillage devient inutile, et une utilisation simple des noyaux Sobel pour extraire l'élément caractéristique aurait été suffisante.

Pour notre projet, la reconnaissance de membres ne se fait pas forcément que sur fond uniforme. En effet, une main peut être aussi bien détectée sur fond blanc que sur fond bruité (herbe, cailloux, textures naturelles, ...). Pour une telle détection, les textures apportent des informations d'image inutiles pour notre objectif. D'où l'utilité d'un seuillage préliminaire.

- Le seuillage effectué et les opérations morphologiques appliquées sont traitées sur des images binaires, donc insensibles au bruit. Le filtre Laplacien a un avantage et inconvénient majeurs: Il permet d'obtenir des contours exclusivement fermés, mais reste extrêmement sensible au bruit.

Ainsi, grâce à l'opérateur Laplacien, tous les contours restants sont fermés, qui facilitera la partie suivante, consistant à construire les cadres englobants.

Une fois les traitements morphologiques effectués, l'image contient des zones fermées que nous nommerons imagettes. Chaque imagette contient des informations de forme, d'aire, de couleur, qui nous seront utiles pour la suite de notre programme. L'objectif est donc d'isoler chacune de ces formes afin de les analyser par la suite.
Pour y parvenir, nous construisons donc des cadres englobants, dont la hauteur et la largeur sont déterminées par la diagonale de la forme. Cependant, les opérations morphologiques étant réalisées de manière automatique, il est possible que tous les pixels parasites ne disparaissent pas complètement, créant ainsi de nombreux cadres englobants non pertinents, et couteux en calculs.
Afin de pallier à ce problème, nous avons donc effectué un filtrage sur les aires des rectangles, ne prenant en compte que les aires supérieures à 500 pixel, nous réduisant le nombre d'imagettes à 5 maximum par image.

## II.3 Reconnaissance des imagettes

### Template Matching

Chaque imagette trouvée doit maintenant être comparée avec une image de référence, afin de déterminer si la zone détectée correspond bien à une main. Nous sommes donc partis dans un premier temps sur l'algorithme statistique du Template Matching. Facile d'approche, il permet de corréler une image avec une image de référence (plus petite) et d'en ressortir un score. Si ce score est assez élevé, la forme détectée est considérée comme reconnue.
Pour notre projet, le Template Matching devient un algorithme très puissant pour des photos standardisées nécessitant un même plan pour chaque membre de chaque personne. C'est sur ce type d'image que nous avons testé notre programme.
Malheureusement, dès que l'environnement devient variable, cet algorithme perd en robustesse. En effet, le Template Matching ne se basant que sur une seule image de référence, les rotations et changements d'échelle du membre ne sont pas pris en compte.

### Axes d'optimisation

Deux possibilités s'offrent comme optimisation de reconnaissance statistique:

- Constituer une base de donnée proposant une position spécifique pour chaque membre et les classifier. Chacun des élément parcourt un a un dans l'image et le programme associe l'élément en fonction du nombre d'élément classifié le plus présent. Cela nous permettrait ainsi, dans une optique plus lointaine, de partir sur un modèle de dictionnaire de type Sac-De-Mots.
- Pour chaque imagette, récupérer des éléments caractéristiques stockées dans des matrices comme valeurs propres. C'est le principe de l'algorithme du Principal Component Analysis (PCA). Il reste cependant sensible au changement d'échelle, mais est moins couteux en calculs que le Sac-de-Mots.

# III Fonctions utilisées

**Seuillage**

**imread() :** charge dans une matrice les pixels de l'image voulue

**imshow() :** affiche l'image

**namedWindow() :** créé une nouvelle fenêtre pour l'affichage

**waitKey() :** instaure un temps d'attente entre les images. waitKey(0) met le programme en pause jusqu'à l'action de l'utilisateur sur une touche du clavier

**resize() :** redimensionne l'image

**cvtColor() :** convertit le domaine de couleur d'une image dans le domaine voulu

**Opérations Morphologiques**

**dilate() :** à partir d'un élement structurant, permet de dilater la zone du pixel suivant le masque

**erode() :** même principe que dilate, mais en érodant la zone autour du pixel

**Laplacian() :** applique un filtre laplacien

**Cadres englobants**

**findContours() :** fonction stockant dans un vecteur des cellules de vecteurs correspondant au nombre de contours fermes. Les cellules contiennent chacune un vecteur de toutes les coordonnées des contours fermés

**approxPolyDP() :** permet d'approximer, a partir des contours fermes détectés avec findContours, la meilleure approximation polynomiale de la forme géométrique. Permet une plus grande rapidité de calcul pour la création des rectangles, et de trouver la plus grande diagonale.

**boundingRect() :** permet de créer un rectangle autour du contour fermé en fournissant le pixel supérieur gauche et le pixel inférieur droit

**drawContours() :** permet d'afficher les contours recherchés. Dans notre projet, nous affichons les contours fermés trouvés dont l'aire est supérieure a 500 pixels

**rectangle() :** permet de dessiner des rectangles. Dans notre projet, ces rectangles sont dessinés à partir des coordonnées des cadres déterminés auparavant.

**Template Matching**

**matchTemplate() :** permet d'effectuer un template matching

**normalize() :** normalise les valeurs trouvées pour le score de corrélation par la surface de l'image

**minMaxLoc() :** permet de trouver les valeurs maximales ou minimales (dépend du modèle de template matching choisi) des différents scores de corrélation.

# IV Conclusion

Ce projet nous a permis de nous familiariser avec l'environnement du C++, langage de programmation que nous ne connaissions pas jusqu'alors. Nos connaissances respectives en traitement d'image nous ont permis d'établir un algorithme efficace, malgré l'optimisation que nous pourrons apporter sur la partie reconnaissance statistique, avec l'extension au sac de mots ou au PCA.
OpenCV étant une bibliothèque Open-Source, nous avons trouvé des aides sur le tutoriel fourni pour certaines parties de nos algorithmes, notamment pour le calcul d'histogrammes, utilisé pour des tests de vérification. Ces tutoriels nous ont aussi orienté sur la création des cadres englobants et l'utilisation de la fonction matchTemplate.
Ainsi, ce module a pu apporter une double approche de la programmation dans le domaine de l'image: l'apprentissage du langage, et la familiarisation avec l'Open-Source, qui se développe de plus en plus en entreprise.