Master Informatique

Reconnaissance des mouvements de la main

Rapport

en vue de la validation de l'UE Initiation à la recherche

Étudiants : Victor DALLÉ
Claire KURTH

Encadrante : Madame BOLTCHEVA

Décharge de responsabilité

L'Université de Lorraine n'entend donner ni approbation ni improbabtion aux opinions émises dans ce rapport, ces opinions devant être considérées comme propres à leurs auteurs.

Remerciements

Table des matières

Introduction

De nos jours, la vision par ordinateur est un domaine en plein essor. La reconnaissance de gestes fait partie intégrante de ce domaine et à ce titre, incarne une révolution dans la manière dont les utilisateurs interagissent avec les systèmes informatiques. Cette technologie est en effet en train de transformer la façon dont nous interagissont avec les machines. Cette avancée offre des opportunités novatrices dans des domaines tels que l'interaction entre l'homme et la machine, la réalité augmentée ou encore l'accessibilité numérique. De nombreuses techniques existent déjà pour permettre la détection des mains. MediaPipe de Google [src] utilise le machine learning pour entrainer un modèle qui détecte les segments composant la main. D'autres travaux ont été réalisés comme ceux de l'équipe du professeur Kalpana Joshi [src] qui utilise les angles formés entre 2 doigts pour détecter la forme de la main (nombre de doigts, main ouverte ou fermée).

Contrairement aux interfaces traditionnelles basées sur le clavier et la souris, la reconnaissance des gestes permet aux utilisateurs de communiquer plus simplement avec les ordinateurs. Ils peuvent désormais avoir recours à leurs mains ou à leur corps pour contrôler les applications, ou encore naviguer dans des environnements virtuels. Cette approche favorise une expérience utilisateur plus immersive et ergonomique, ouvrant ainsi de nouvelles perspectives dans des domaines variés tels que le divertissement interactif, l'éducation, ou encore la médecine. Comme dit précédemment, la reconnaissance des mouvements joue un rôle crucial dans l'accessibilité numérique en permettant à des personnes porteuses d'un handicap physique ou moteur de pouvoir communiquer et d'interagir avec des outils numériques plus facilement. En effet, celà permet de passer outre les obstacles liés à l'utilisation des outils traditionnels (tels que le clavier, la souris, la télécommande ...) grâce à la simple utilisation de mouvements du corps. Cette nouvelle manière d'interagir avec un système numérique est déjà utilisée dans plusieurs domaines notamment le sport avec des applications de coaching personnel qui permettent de suivre les mouvements de l'utilisateur et ainsi lui donner des conseils pour améliorer sa technique, ou encore sa posture. Ce nouveau concept d'interraction permet également de pouvoir contrôler des appareils tels que des téléviseurs où par un simple geste, nous pouvons par exemple gérer le son ou changer de chaîne.

Dans ce contexte, ce projet vise à se questionner vis-à-vis d'un système de reconnaissance des mouvements de la main à l'aide d'un classifieur classique Haar-cascade et de le comparer avec un système de reconnaissance de la main sans classifieur. L'objectif principal est de concevoir un système capable de détecter et de classifier différents gestes de la main effectués par l'utilisateur, tels que le poing fermé, ou alors la main ouverte avec un certains nombre de doigts levés. Ces gestes seront ensuite associés à différentes actions telles que le lancement d'applications ou encore l'ouverture de sites web. Pour réaliser ce projet nous utiliserons principalement la bibliothèque OpenCV. Nous parlerons dans un premier temps plus en détail des techniques de reconnaissance de gestes actuellement utilisées, puis nous expliciterons les notions techniques ainsi que les méthodes que nous exploiterons. Nous verrons ensuite comment nous avons implémenté la reconnaissance de la main sans le classifieur classique Haar-Cascade puis avec ce classifieur.

1 Rappel du sujet et encadrement

1.1 Rappel du sujet

Le but de ce projet est d'implémenter un système de reconnaissance des mouvements de la main à l'aide d'un classifieur classique Haar-cascade. Le système doit reconnaître le geste de la main de l'utilisateur (poing, un doigt, deux, trois, quatre,...) et le mapper à différentes tâches telles que le lancement d'applications comme le bloc-notes, la peinture, et l'ouverture de sites web. Le système doit être mis en œuvre avec l'aide de la librairie de "Computer Vision" - OpenCV, comme dans l'article [src]. Une extension possible serait l'implémentation d'un système de détection des mouvements de la tête ou du corps, tout entier.

1.2 Encadrement

2 État de l'art

2.1 Article de départ

2.2 Médiapipe

Médiapipe est un framework open-source développé par Google permettant de construire des pipelines de traitement de données multimédia. Il propose des solutions pour la détection de la main, du visage ou encore de la pose. Il est basé sur des modèles de machine learning notamment grâce à de l'apprentissage via des réseaux de neurones.

[Image de la main détectée par Médiapipe]

2.3 classifieur classique Haar-cascade

Le classifieur Haar-cascade est une méthode de détection d'objets dans une image introduit par Paul Viola et Michael Jones en 2001 [src]. Il est basé sur l'utilisation de caractéristiques de type Haar. Ces caractéristiques sont des fenêtres de taille fixe qui sont déplacées sur l'image et qui permettent de calculer la différence de luminosité entre les pixels de la fenêtre. Ces caractéristiques sont ensuite utilisées pour entraîner un classifieur qui permet de détecter des objets dans une image.

Les classifieurs Haar-cascade sont utilisés pour la détection de visages, de voitures, de plaques d'immatriculation, de piétons, de mains ou de tout autres objets. Ils sont très utilisés dans le domaine de la vision par ordinateur et sont très efficaces pour la détection d'objets dans une image.

3 Reconnaissance de la main avec classifieur classique Haar-cascade

3.1 Entrainer un classifieur Haar-cascade

Pour entrainer un classifieur Haar-cascade, il faut tout d'abord collecter des images positives et négatives. Les images positives sont des images contenant l'objet que l'on souhaite détecter, tandis que les images négatives sont des images ne contenant pas l'objet. Il faut ensuite générer des fichiers de descriptions des images positives et négatives. Ces fichiers contiennent les coordonnées des objets à détecter dans les images positives. Enfin, il faut entrainer le classifieur à l'aide de ces fichiers de descriptions.

L'entrainement du classifieurs en lui-même se fait grâce à ce que l'on appelle des "features". Ces dernières ont été introduites par Viola et Jones en 2001 (Fig. 1). Par la suite, d'autres features ont été ajoutées (Fig. 2) afin d'améliorer la détection d'objets dans une image.

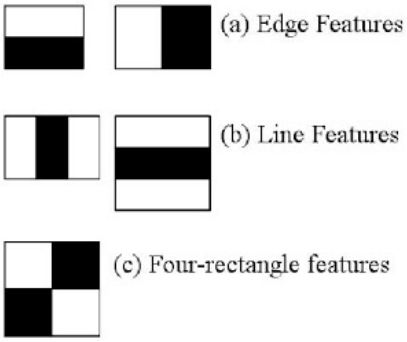

FIGURE 1 – Features de Haar comme utilisées par Viola et Jones.

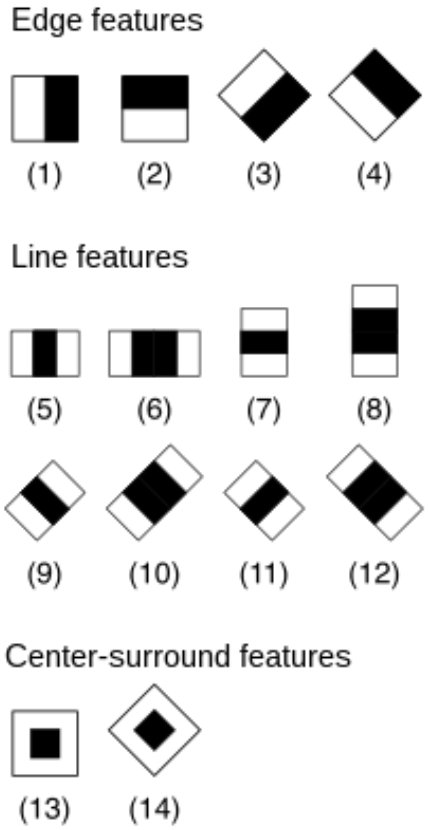

FIGURE 2 – Features de Haar supplémentaires.

Ces features sont des caractéristiques de l'objet que l'on souhaite détecter, ce sont des patterns de pixels qui permettent de distinguer l'objet des autres éléments de l'image.
Il existe différents types de patterns :

— Les "edges" : ce sont des patterns qui permettent de détecter les contours de l'objet.
— Les "lines" : ce sont des patterns qui permettent de détecter les lignes de l'objet.
— Les "center-surrounder" : ce sont des patterns qui permettent de détecter les changements d'intensité entre le centre d'une région rectangulaire et le reste de la région. Cela permet de détecter des objets de forme particulière.

Pour détecter ces patterns, l'algorithme utilise des fenêtres de taille fixe qui sont déplacées sur l'image. Ces fenêtres permettent de calculer la différence de luminosité entre les pixels de la fenêtre. Ces différences de luminosité sont ensuite utilisées pour déterminer si le pattern est présent dans l'image.

Ces features sont ensuite utilisées pour entrainer un classifieur qui permet de détecter l'objet dans une image. Le classifieur est entrainé à l'aide d'un algorithme de machine learning tel que AdaBoost qui permet de déterminer les features les plus pertinentes pour la détection de l'objet.

AdaBoost est un algorithme d'apprentissage supervisé qui permet de construire un classifieur fort à partir de plusieurs classifieurs faibles. Au début, chaque élément de la base de données à le même poids. L'algorithme va ensuite sélectionner un classifieur faible (par exemple, un arbre de décision simple) qui performe légèrement mieux que l'aléatoire. Ce classifieur va être utilisé pour prédire les éléments de la base de données. Les exemples mal classés reçoivent un poids plus élevé, tandis que les exemples correctement classés reçoivent un poids plus faible. Ainsi, les exemples difficiles à classer ont plus d'influence sur la formation du classifieur final. Les poids des classifieurs faibles sont déterminés en fonction de leur précision relative. Les classifieurs les plus précis ont un poids plus élevé. Enfin, le classifieur final est une combinaison linéaire des classifieurs faibles pondérés par leur précision relative.

Src : https ://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm

3.2 Nos entrainements

3.2.1 Avec une base de données d'images de mains

Méthodologie

Pour notre entrainement, nous avons collecté 10 000 images négatives (Fig. 3) et 5 000 images positives de mains (Fig. 4). Nous avons dû pour les images positives, annoter les images (dans un fichier texte) afin de donner le nombre de mains présentes et les coordonnées de la main dans l'image (Fig. 5). Ensuite, grâce à ce fichier et à OpenCV, nous avons généré un fichier vec qui contient les informations des images positives. Nous avons ensuite entrainé le classifieur à l'aide de ce fichiers.

FIGURE 3 – Exemple d'image négative

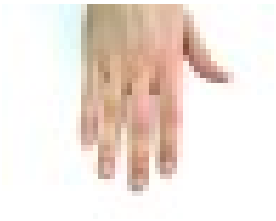

FIGURE 4 – Exemeple d'image positive

```
positives\Hand_0000002.jpg 1 0 0 50 38
positives\Hand_0000003.jpg 1 0 0 50 38
positives\Hand_0000004.jpg 1 0 0 50 38
```

FIGURE 5 – Exemple de fichier de description

Nous sommes ensuite passés à l'entrainement. Nous avons utilisé pour celà OpenCV qui propose un programme pour entrainer un classifieur Haar-cascade. Nous avons testé plusieurs cas : avec 5, 10, 15 et 20 étapes, avec des profondeurs d'arbres maximum différentes, avec plus d'images positives que négatives et inversement. Enfin, avons aussi testé le nombre de "Faux positif" maximum autorisé, c'est-à-dire le nombre maximum d'images négatives mal classées autorisées. Á la fin de l'entrainement, nous récupérons un fichier xml qui sera ensuite utilisé pour la détection.

Faire un tab récap pour les heures

Test	Nombre d'étapes	Profondeur d'arbre	Nombre de faux positifs	Nombres d'heures	Commentaires

JSP SI ON METS UN EXEMPLE DES COMMANDES ? ?

Résultats

Nous voulions obtenir un classifieur qui détecte la main dans une image comme suivant :

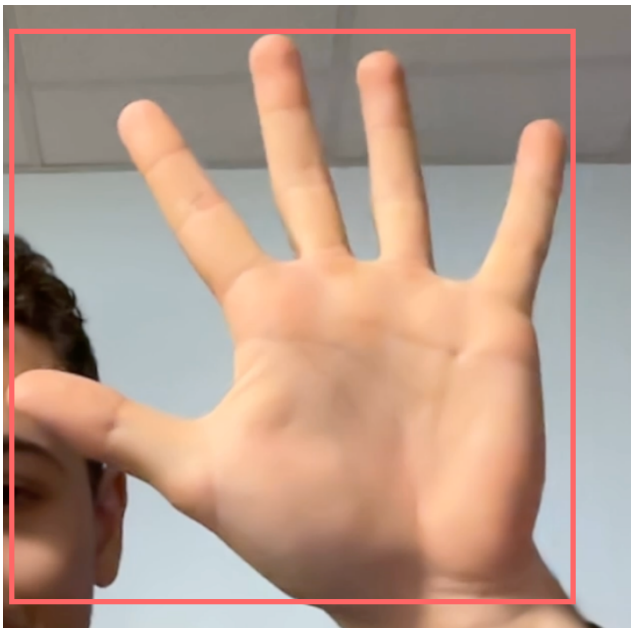

FIGURE 6 – Résultat attendu

A COMPLETER

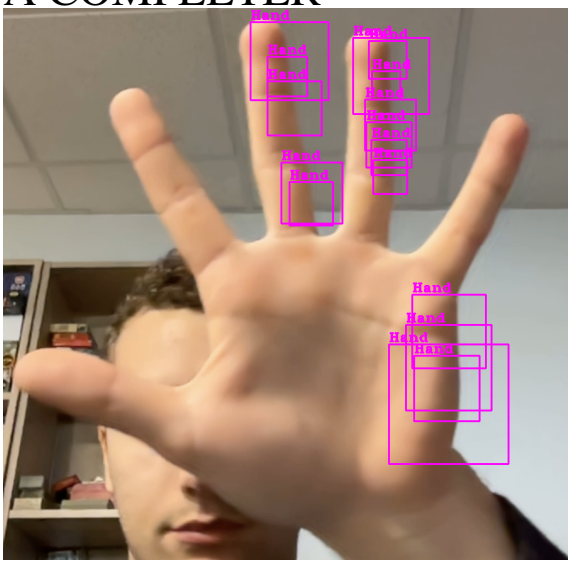

FIGURE 7 – Test 1

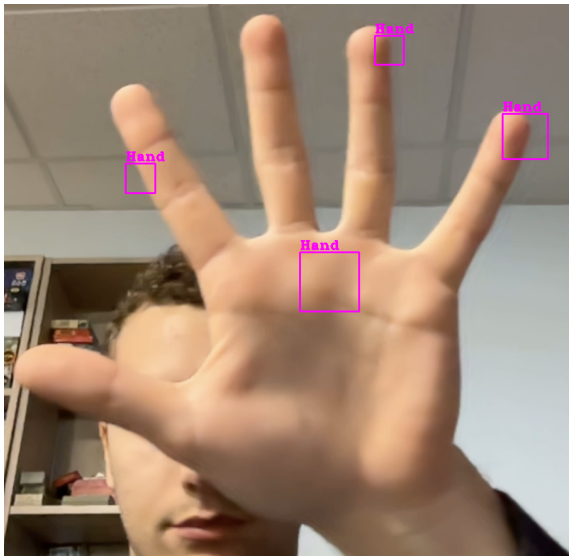

FIGURE 8 – Test 2

FIGURE 9 – Test 3

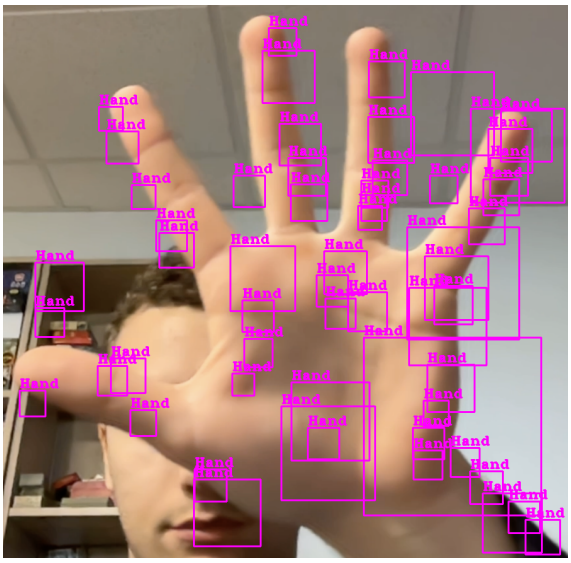

FIGURE 10 – Test 4

Conclusion

Les résultats ne sont pas entièrement satisfaisant. En effet, le classifieur ne détecte pas toute la main mais seulement plusieurs parties de la main. Celà est certainement dû au fait que les images positives ne contienent que la main sans "background" et que les features utilisées ne sont pas suffisamment pertinentes pour détecter la main dans une image.

3.2.2 En créant nous même des images positives

Méthodologie

N'ayant pas obtenus de résultats satisfaisants avec la base de données d'images de mains, nous avons décidé de créer nous même des images positives. Pour cela, nous avons utilisé la librairie OpenCV pour ajouter une image de main aux images positives (Fig. 11). Nous avons ensuite généré les fichiers de descriptions des images positives et entrainé le classifieur à l'aide de ces fichiers.

FIGURE 11 – Exemple d'image positive créée à partir d'une image négative

Comme précédemment, nous avons tester différents paramètres pour l'entrainement du classifieur.

TABLEAU

Résultats

TO DO

Conclusion

Les résultats ne sont pas beaucoup plus satisfaisant. On obtient les mêmes résultats que précédemment.

Il faudrait essayer de prendre une base de données de mains contenant des background différents et annoter nous mêmes les images pour voir si celà améliore les résultats.

4 Reconnaissance du mouvement de la main sans classifieur

4.1 Détection de la main

4.1.1 Méthodologie

Pour réussir à détecter la main sans classifieur, nous avons dû tester plusieurs méthodes. En effet, plusieurs paramètres interviennent afin d'avoir une détection de la main optimale.

On a dû jouer sur plusieurs paramètres :

— Le format de la couleur de l'image : GreyScale ou HSV
— Flou : avec ou sans, quel type de flou (Gaussien ou Bilatéral)
— Thresholding : pour binariser l'image. Il y avait là plusieurs paramètres possibles : le seuil et le type de thresholding (binaire, binaire inversé, tronqué, to zero, to zero inversé)
— Contours : pour détecter les contours de la main. Là aussi, plusieurs paramètres possibles.

4.1.2 Expériences et résultats

Pour trouver les meilleures paramètres, nous avons testé plusieurs combinaisons de paramètres, avec ou sans flou, avec ou sans Canny ...

Tout d'abord, nous avons essayé en essayant de mettre l'images en gris, puis de flouter l'image avec un flou gaussien, puis de binariser l'image avec un thresholding binaire. Enfin, nous avons utilisé Canny pour détecter les contours de la main et nous avons tracer le convexe hull de la main. Les résultats sont plutôt bons puisque nous avons capturé l'essentiel de la main même si il manque une grosse partie du pouce.

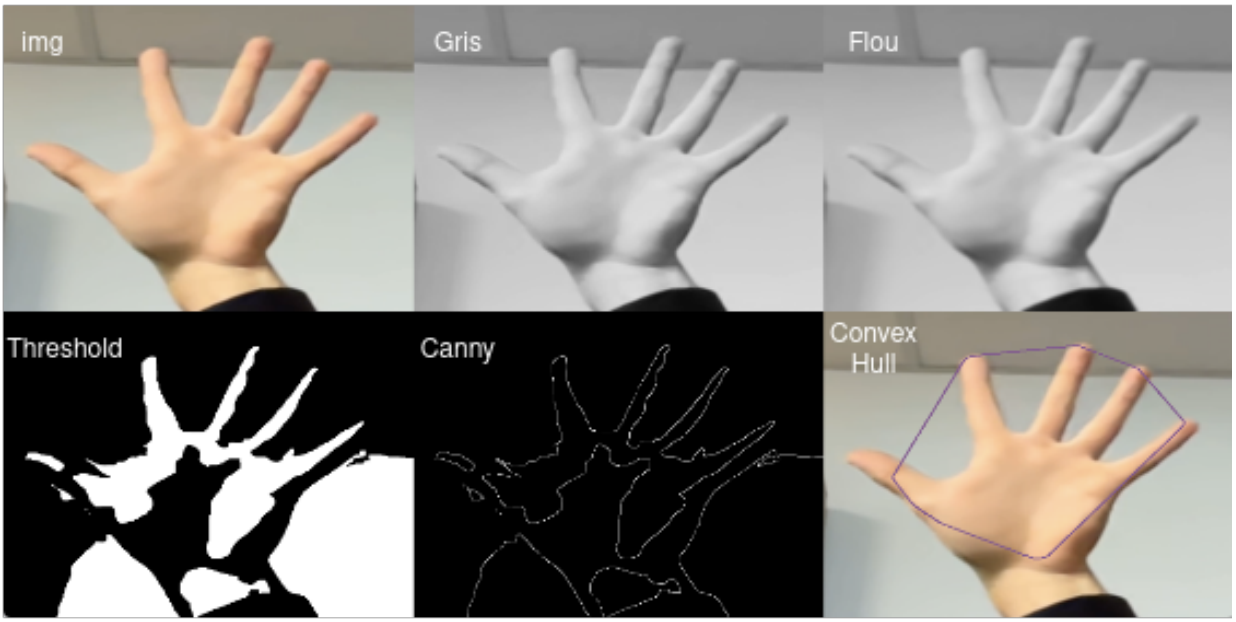

FIGURE 12 – Résultat de la détection de la main :
Thresholding : seuil entre 200-255, TRESH_BINARY
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

Nous avons aussi essayé sans flou, les résultats étaient moins bons puisque nous pouvons voir plusieurs convex hulls pour une seule main.

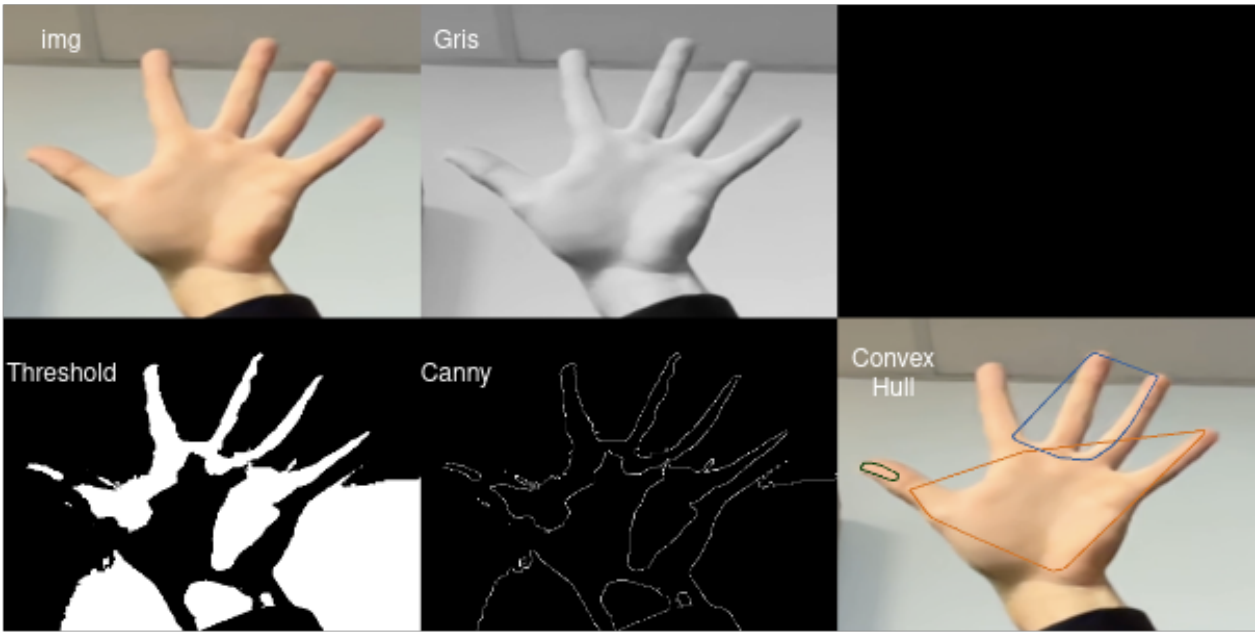

FIGURE 13 – Résultat de la détection de la main :
Thresholding : seuil entre 198-255, TRESH_BINARY
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

Nous avons ensuite essayé sans flou et sans Canny. Les résultats sont encores moins bons puisque nous avons plusieurs convex hulls voir même certains hors de la main.

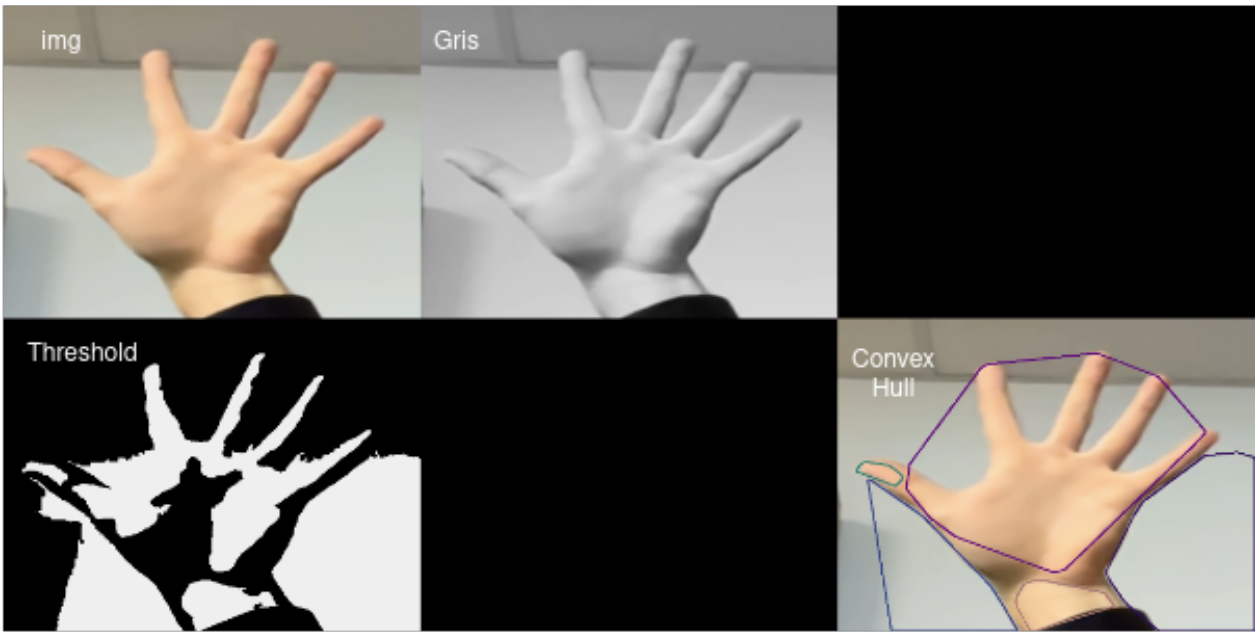

FIGURE 14 – Résultat de la détection de la main :
Thresholding : seuil entre 196-239, TRESH_BINARY
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

Les résultats ne sont pas exceptionnels mais nous nous attendions à ce qu'ils soient meilleurs en passant l'image en HSV plutôt qu'en GreyScale.

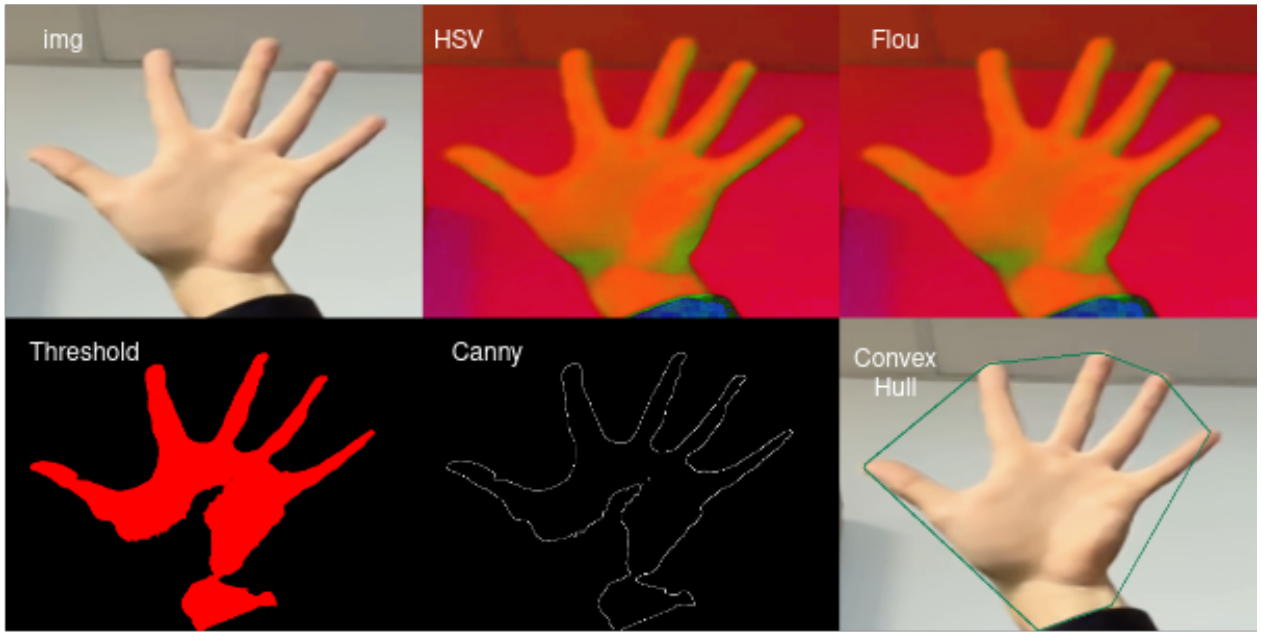

FIGURE 15 – Résultat de la détection de la main :
Thresholding : seuil entre 223-255, TRESH_BINARY
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

On voit clairement ici que les résultats sont meilleurs puisque nous avons bien capturé l'ensemble de la main. De plus, même sur une image de mauvaise qualité, en jouant sur les paramètres, nous avons réussi à capturer la main.

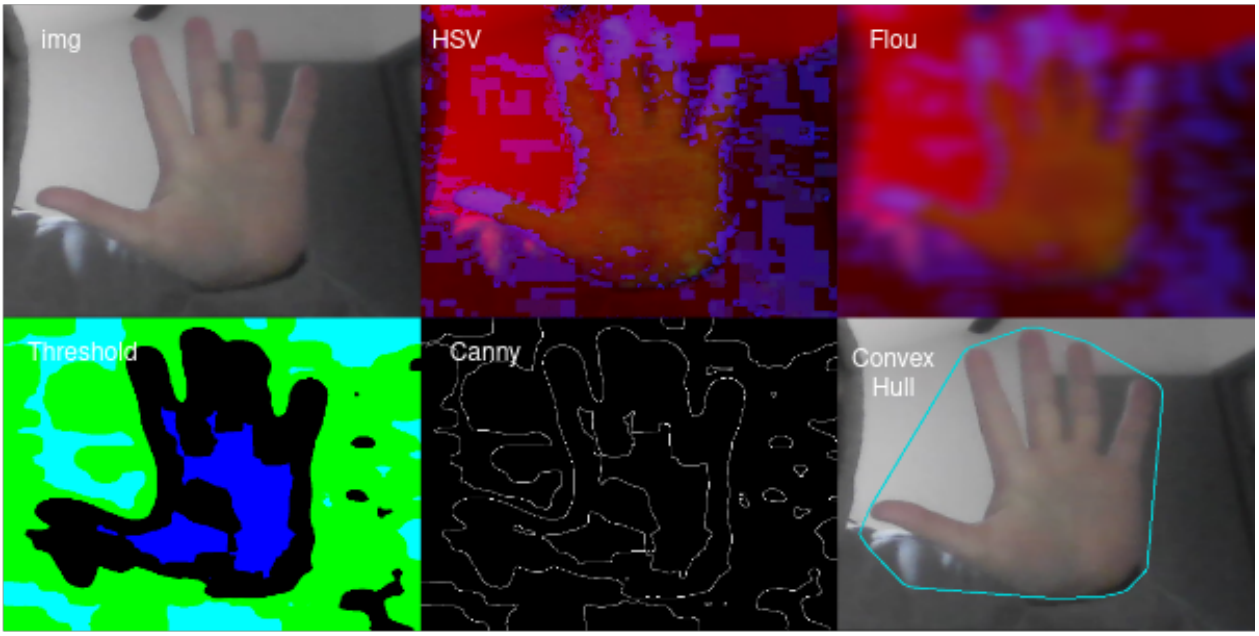

FIGURE 16 – Résultat de la détection de la main :
Flou : X = 41, Y = 27, Sigma = 22
Thresholding : seuil entre 11-255, TRESH_BINARY
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

Enfin, nous avons essayé sans passer par la fonction thresholding et canny mais en créant nous même un masque en fonction de la saturation, de la valeur et de la teinte.

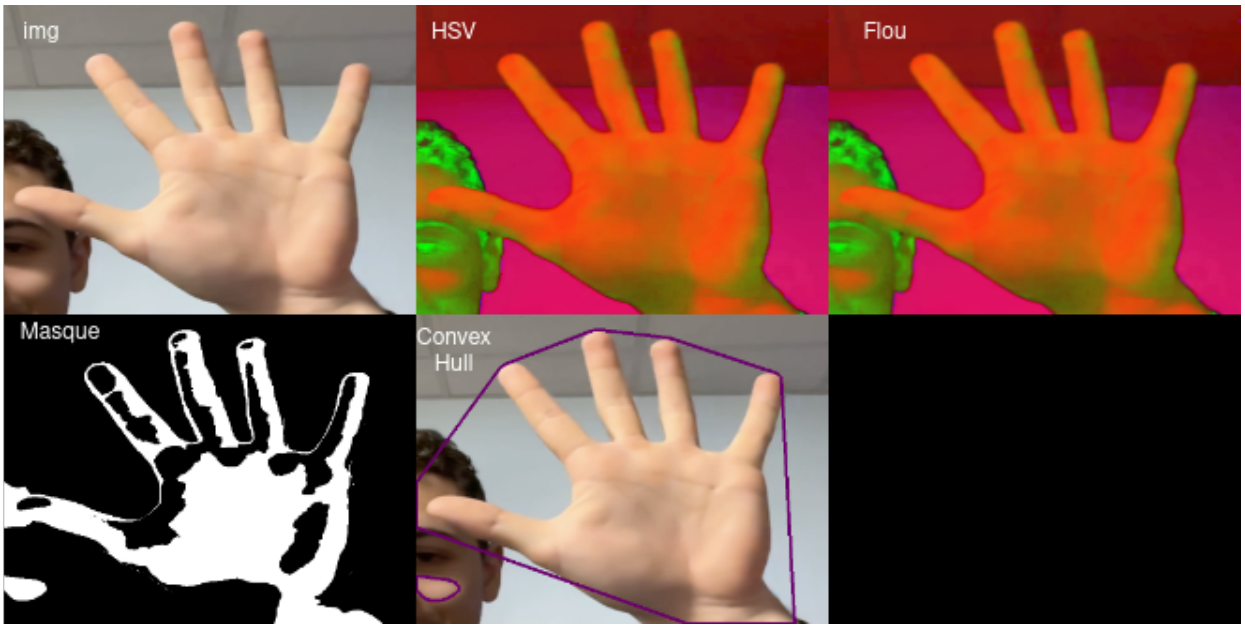

FIGURE 17 – Résultat de la détection de la main : Hue : 0 - 94, Saturation : 37 - 180, Value : 145 - 238

Les résultats sont bons. Cependant, les résultats obtenus sont très dépendant des valeurs des paramètres.

4.1.3 Conclusion

Même si les résultats en passant l'image en HSV sont concluants, il reste encore des améliorations à apporter puisque la détection est très dépendante des paramètres utilisés. Il faudrait donc trouver une méthode pour que le programme puisse trouver les meilleurs paramètres pour la détection de la main.

4.2 Détection des doigts

4.2.1 Sans Convexe Hull

Pour la détection des doigts, nous avons créer un masque ayant la forme d'un doigts que nous passons ensuite sur l'image. Nous déplaçons ensuite ce masque sur l'image et calculons le nombre de pixels blancs dans le masque. Si ce nombre est entre un certains seuil, alors nous considérons que le doigt est présent.

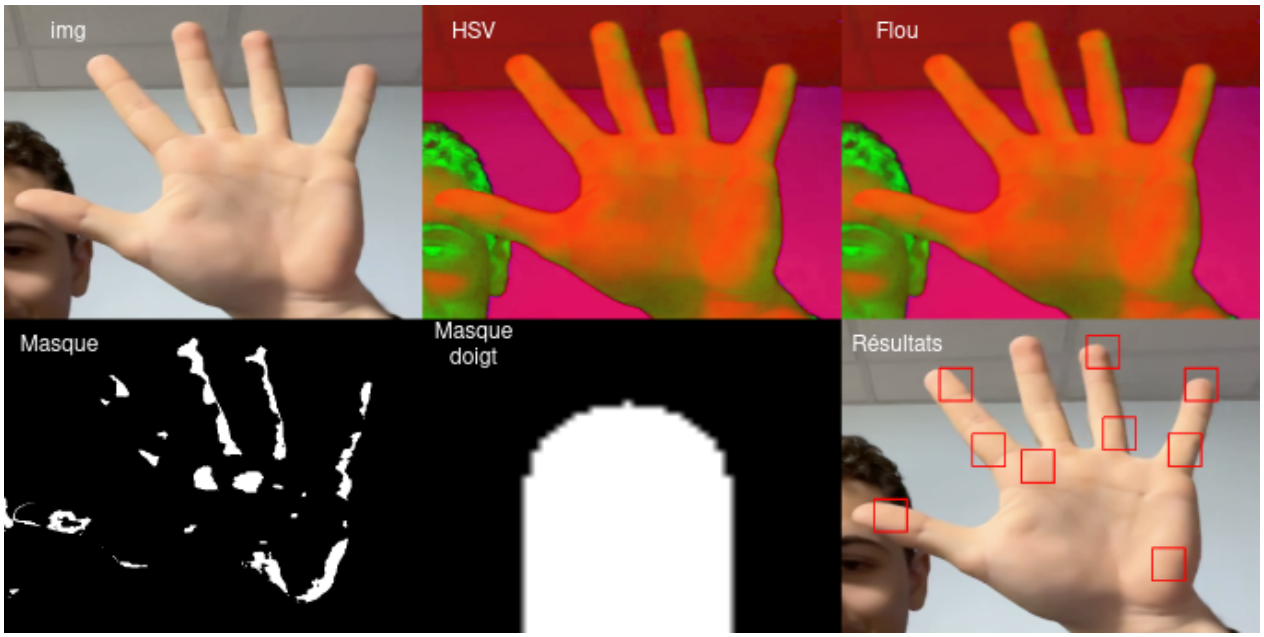

FIGURE 18 – Détection des doigts sans Convexe Hull. Hue : 8 - 67 ; Saturation : 97 - 128 ; Value : 185 - 239 ; Seuil : 73 - 100

5 Fusion des deux systèmes

5.1 Méthodologie

Nous avons un classifieur d'une part qui détecte une portion de la main et d'autres part un système qui permet de détecter la main si on a bien configuré les paramètres. Nous allons donc fusionner ces deux systèmes : si le classifieur détecte une portion de la main, on utilise la plus grande portion de la main détectée afin de récupéré tout d'abord la couleur de la main puis

5.2 Expériences et résultats

5.3 Conclusion

Conclusion

Annexes

Bibliographie

Glossaire

Déclaration sur l'honneur contre le plagiat

Résumé