



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	1
1 Rappel du sujet et encadrement	2
1.1 Rappel du sujet	2
1.2 Encadrement	2
2 État de l'art	3
2.1 Article de départ	3
2.2 Médiapipe	3
2.3 classifieur classique Haar-cascade	3
3 Reconnaissance de la main avec classifieur classique Haar-cascade	5
3.1 Entrainer un classifieur Haar-cascade	5
3.2 Nos entrainements	6
3.2.1 Avec une base de données d'images de mains	6
3.2.2 En créant nous même des images positives	9
4 Reconnaissance du mouvement de la main sans classifieur	12
4.1 Détection de la main	12
4.1.1 Méthodologie	12
4.1.2 Expériences et résultats	13
4.1.3 Conclusion	17
4.2 Détection des doigts	17
4.2.1 Sans Convex Hull	17
5 Fusion des deux systèmes	18
5.1 Méthodologie	18
5.2 Expériences et résultats	18
5.3 Conclusion	18
Conclusion	19
Annexes	20
Bibliographie	20

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 interagissons 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 entraîner 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, cela 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'interaction 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

L'article sur lequel nous nous basons, Static Hand Gesture and Face Recognition System **SOURCE** propose un système de reconnaissance de gestes de la main. Ce système est créé à partir d'une image, passée en HSV à laquelle ils ont ajouté un Flou et un Thresholding. Le Thresholding permet de générer une image binaire : chaque pixel est comparé à un seuil, si la valeur du pixel est supérieur à ce seuil le pixel est blanc, sinon il est noir. Ils extraient ensuite les contours avant de tracer le Convex Hull, le polygone de la main. Enfin, ils calculent des "Convexity Defect" c'est-à-dire des points éloignés de points convexes. Dans ce cas ci, les points convexes sont le bout des doigts et donc les defects sont le trou entre 2 doigts. Un defect est donc compté si l'angle entre 2 doigts est supérieur à 90°.

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 (ou features) 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.



FIGURE 1 – Exemple de détection de visage avec un classifieur Haar-cascade

Les classificateurs Haar-cascade sont utilisés pour la détection de visages (Fig. 1), 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 entraîner 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 entraîner le classifieur à l'aide de ces fichiers de descriptions.

L'entraînement 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. 2). Par la suite, d'autres features ont été ajoutées (Fig. 3) afin d'améliorer la détection d'objets dans une image.

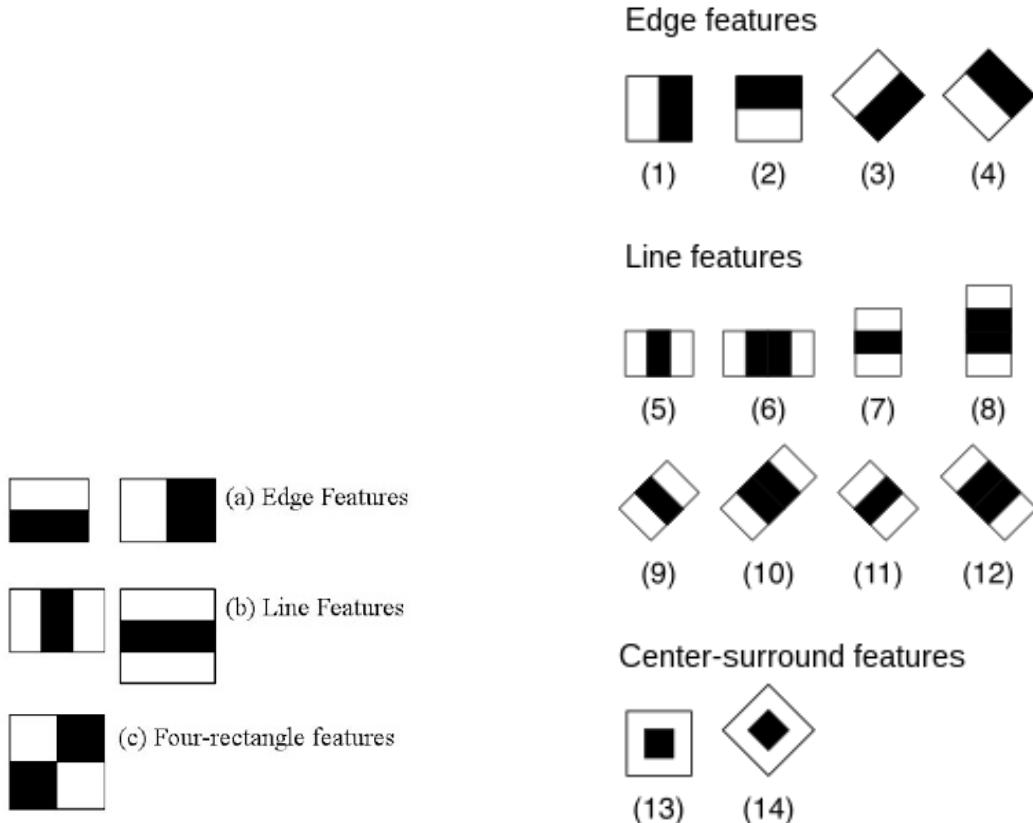


FIGURE 2 – Features de Haar comme utilisées par Viola et Jones.

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 :

FIGURE 3 – Features de Haar supplémentaires.

- 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 entraîner un classifieur qui permet de détecter l'objet dans une image. Le classifieur est entraîné 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 classificateurs faibles. Au début, chaque élément de la base de données a le même poids. L'algorithme va ensuite sélectionner un classificateur faible (par exemple, un arbre de décision simple) qui performe légèrement mieux que l'aleatoire. Ce classificateur 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 classificateur final. Les poids des classificateurs faibles sont déterminés en fonction de leur précision relative. Les classificateurs les plus précis ont un poids plus élevé. Enfin, le classificateur final est une combinaison linéaire des classificateurs faibles pondérés par leur précision relative.

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

3.2 Nos entraînements

3.2.1 Avec une base de données d'images de mains

Méthodologie

Pour notre entraînement, nous avons collecté 10 000 images négatives (Fig. 4) et 5 000 images positives de mains (Fig. 5). 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. 6). Ensuite, grâce à ce fichier et à OpenCV, nous avons généré un fichier vec qui contient les informations des images positives. Nous avons ensuite entraîné le classificateur à l'aide de ce fichiers.



FIGURE 4 – Exemple d’image négative



FIGURE 5 – 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 6 – Exemple de fichier de description

Nous sommes ensuite passés à l’entraînement. Nous avons utilisé pour celà OpenCV qui propose un programme pour entraîner 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. Nous avons également testé différentes valeurs pour le seuil d’acceptation du ratio break. Ce seuil détermine à quel point le modèle continue à apprendre avec précision et quand il doit s’arrêter. Enfin, nous avons testés avec deux modes différents : par défaut et ’ALL’. Le mode par défaut utilisent les features de bases (Fig. 2) tandis que le mode ’ALL’ utilisent les features un peu plus complexes (Fig. 3). À la fin de l’entraînement, nous récupérons un fichier xml qui sera ensuite utilisé pour la détection.

Voici un tableau récapitulatif des différents tests que nous avons effectués :

Test	Nombre de positifs	Nombre de négatifs	Nombre d’étapes	Profondeur d’arbre	Acceptance Ratio Break	Mode	Temps
1	4 000	10 000	7	1	désactivé	Défaut	1h10
2	4 000	10 000	8	1	1.0e-5	Défaut	40min
3	4 000	2 000	10	3	1.0e-5	All	5h
4	4 000	2 000	6	1	1.0e-5	Défaut	4mn

JSP SI ON METS UN EXEMPLE DES COMMANDES ??

Résultats

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

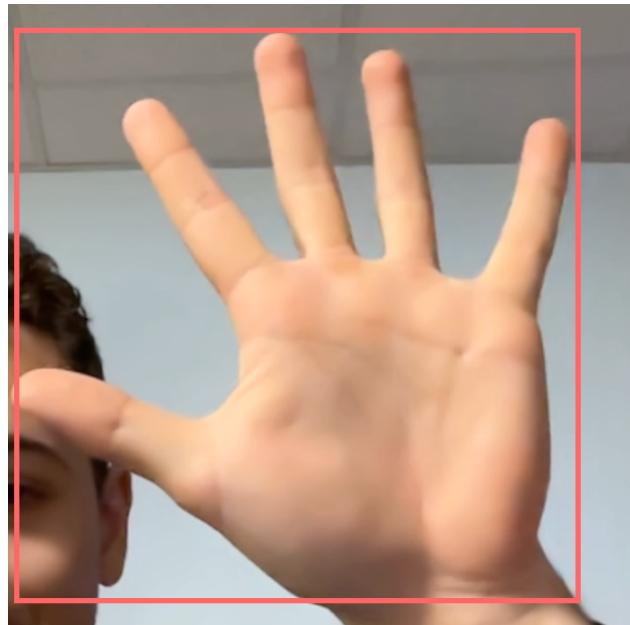


FIGURE 7 – Résultat attendu

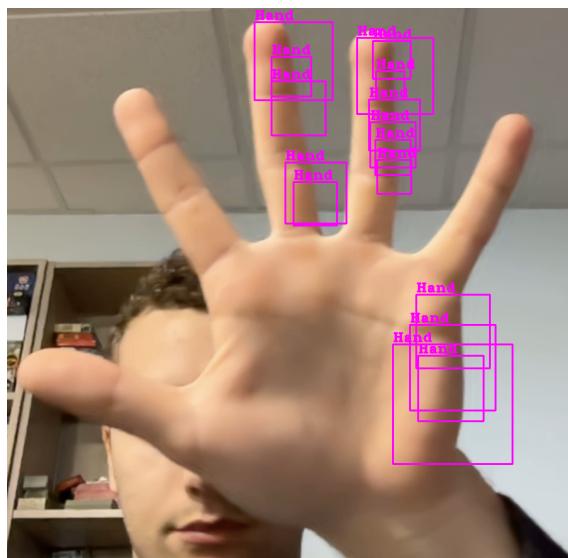


FIGURE 8 – Test 1

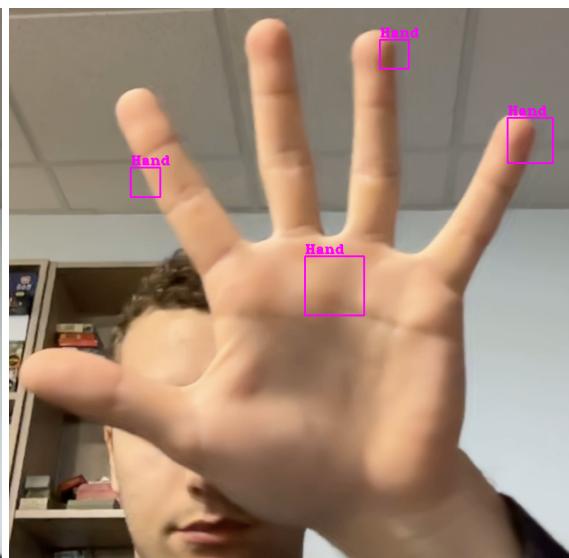


FIGURE 9 – Test 2



FIGURE 10 – Test 3

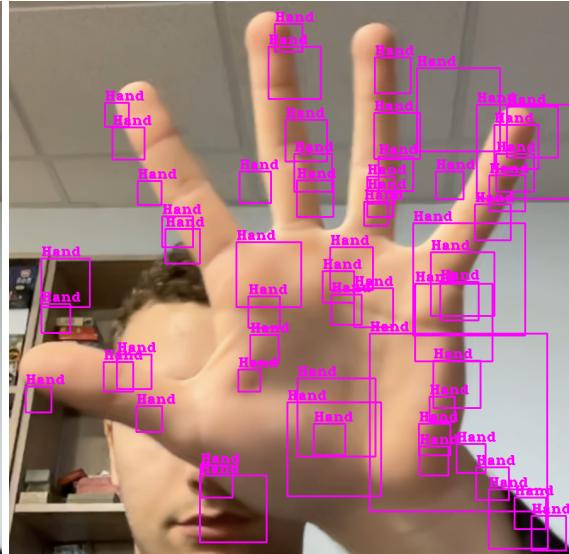


FIGURE 11 – Test 4

Les classifieurs 1 (Fig. 8) et 2 (Fig. 9) ne détectent pas entièrement la main dans l'image, seulement plusieurs portions plus ou moins grandes. Le classifieur 4 (Fig. 11) détecte bien la main mais détecte aussi d'autres objets dans l'image. Le classifieur 3 (Fig. 10) est le moins satisfaisant puisqu'il ne détecte rien du tout, certainement dû à un surapprentissage.

Conclusion

Les résultats ne sont pas entièrement satisfaisant. En effet, les différents classifieurs, lorsqu'ils détectent la main, ne la détectent pas entièrement mais seulement plusieurs parties. Celà est certainement dû au fait que les images positives ne contiennent que la main sans "background".

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. 12). Nous avons ensuite généré les fichiers de descriptions des images positives et entraîné le classifieur à l'aide de ces fichiers.

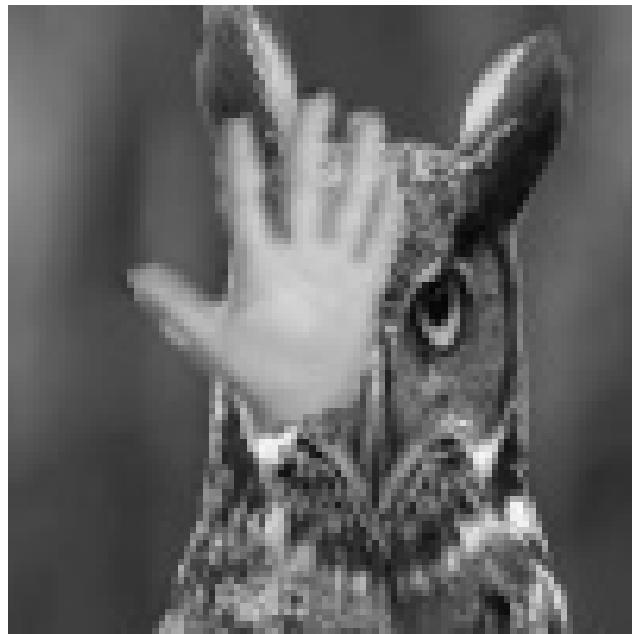


FIGURE 12 – Exemple d'image positive créée à partir d'une image négative

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

Test	Nombre de positifs	Nombre de négatifs	Nombre d'étapes	Profondeur d'arbre	Acceptance Ratio Break	Mode	Temps
5	8 000	6 000	15	1	1.0e-5	Défaut	4h05
6	8 000	6 000	10	1	1.0e-5	Défaut	1h53
7	8 000	6 000	5	1	1.0e-5	Défaut	35min
8	5 000	8 000	9	1	1.0e-5	All	2h49
9	5 000	8 000	5	1	1.0e-5	All	1h25

Résultats



FIGURE 13 – Test 5

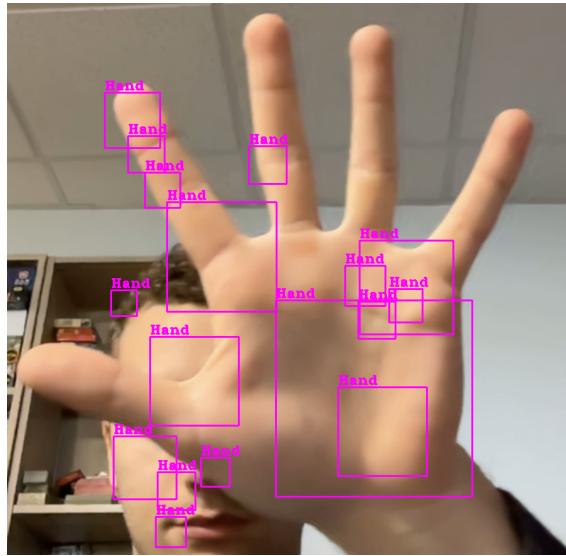


FIGURE 14 – Test 6

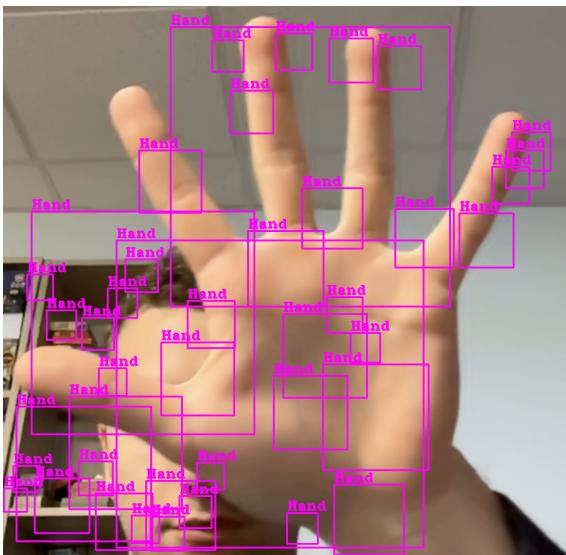


FIGURE 15 – Test 7

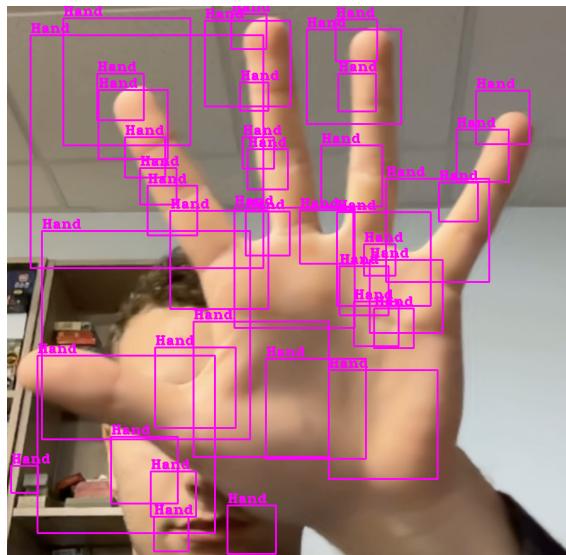


FIGURE 16 – Test 8

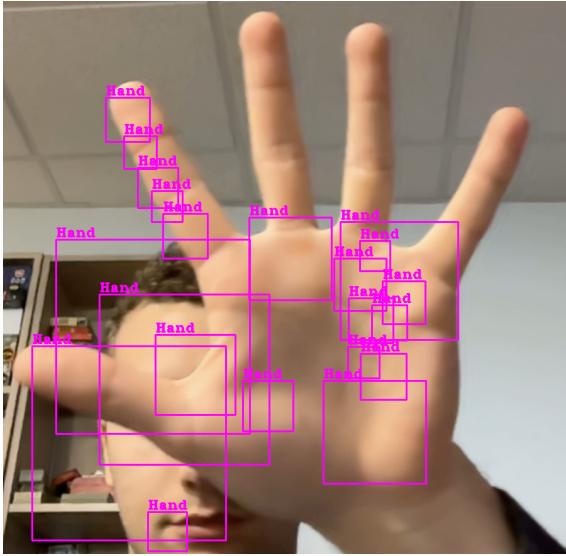


FIGURE 17 – Test 9

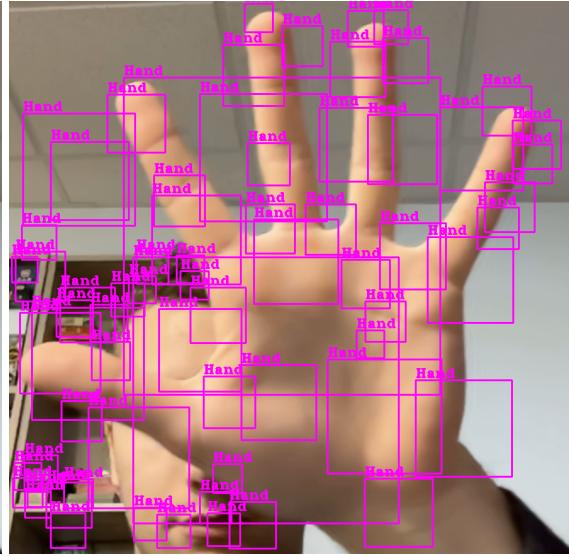


FIGURE 18 – Test 10

Les résultats ne sont pas satisfaisants. Le classifieur 5 (Fig. 13) ne détecte rien du tout, certainement dû à un surapprentissage. Les classifieurs 7 (Fig. 15), 8 (Fig. 16) et 10 (Fig. 18) détectent bien la main mais détectent aussi beaucoup d'autres objets dans l'image. Les classifieurs 6 (Fig. 14) et 9 (Fig. 17) ne détectent pas entièrement la main dans l'image, seulement plusieurs portions plus ou moins grandes. Ils détectent aussi notamment une partie de la bouche.

Conclusion

Les résultats ne sont pas beaucoup plus satisfaisant. On obtient dans l'ensemble, les mêmes résultats que précédemment c'est-à-dire des détections partiels de la main et non la main entières pour les classifieurs les mieux entraînés.

Il faudrait essayer de prendre une base de données de mains contenant des background différents et annoter nous mêmes les images, c'est-à-dire l'emplacement des mains dans l'image, pour voir si celà améliore les résultats.

Pour reconnaître chaque mouvement indépendamment les uns des autres, il faudrait entraîner une cascade par mouvement ce qui n'est pas forcément très pratique.

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 convex 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.

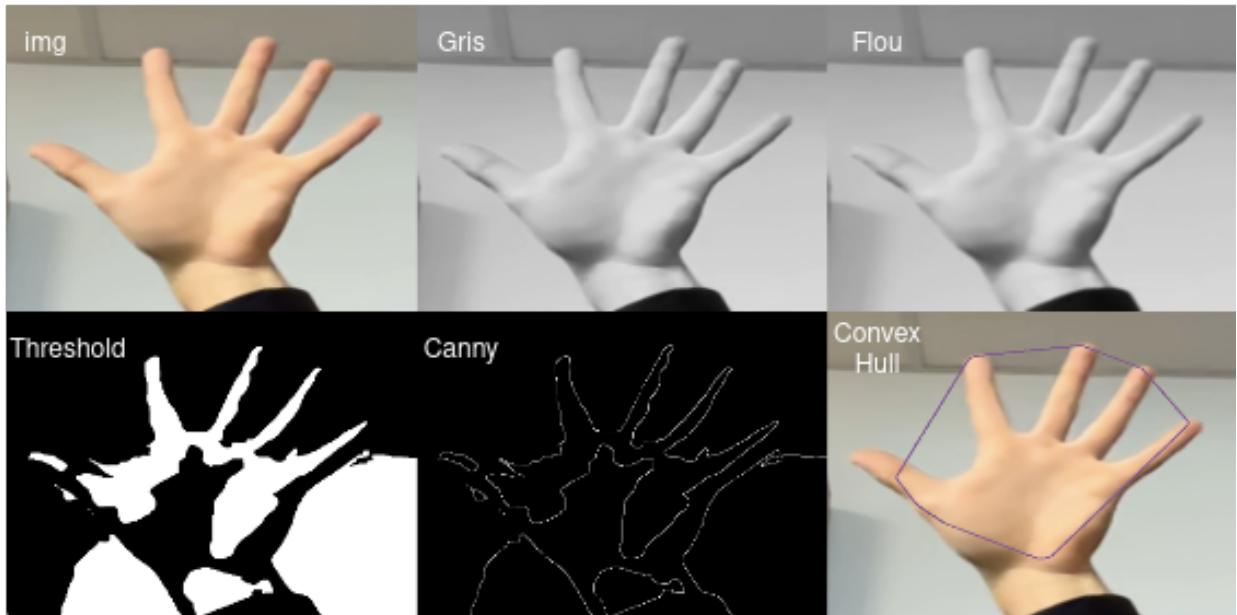


FIGURE 19 – Résultat de la détection de la main : Thresholding : seuil entre 200-255, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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

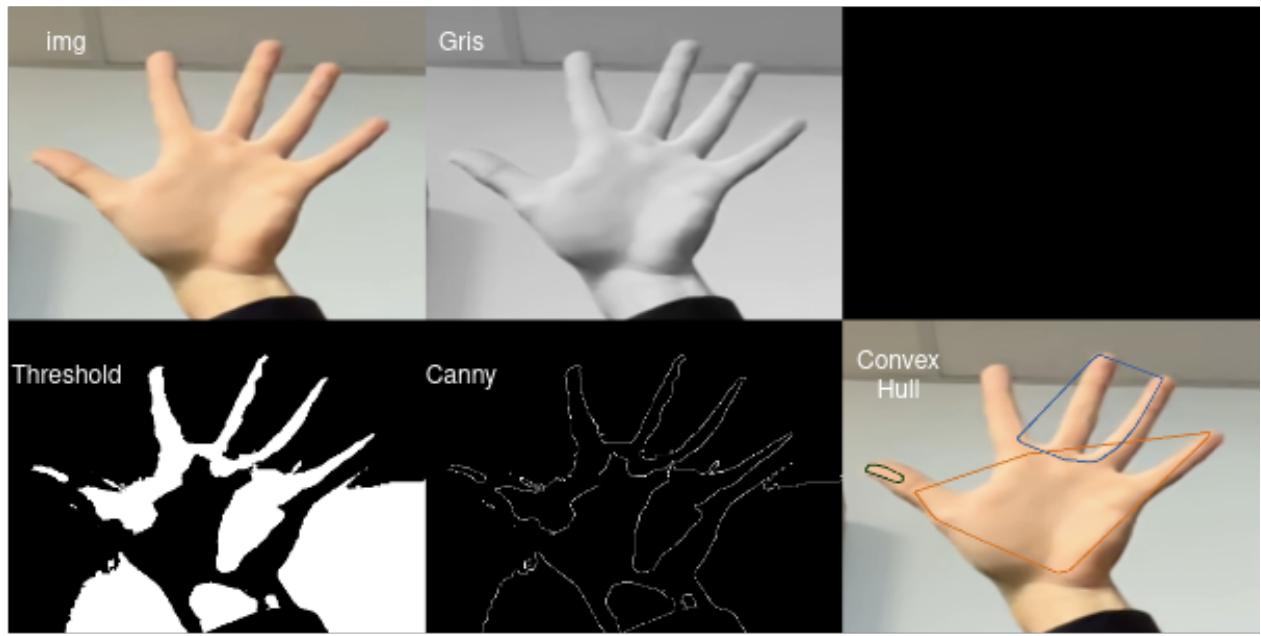


FIGURE 20 – Résultat de la détection de la main : Thresholding : seuil entre 198-255, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

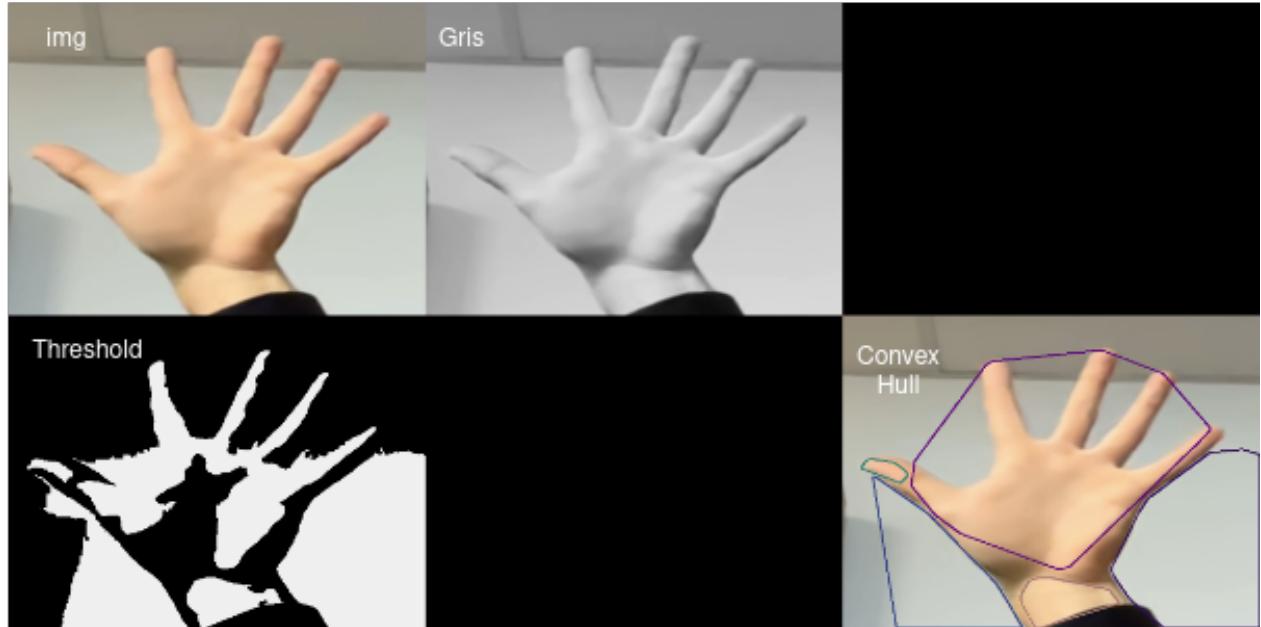


FIGURE 21 – Résultat de la détection de la main : Thresholding : seuil entre 196-239, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

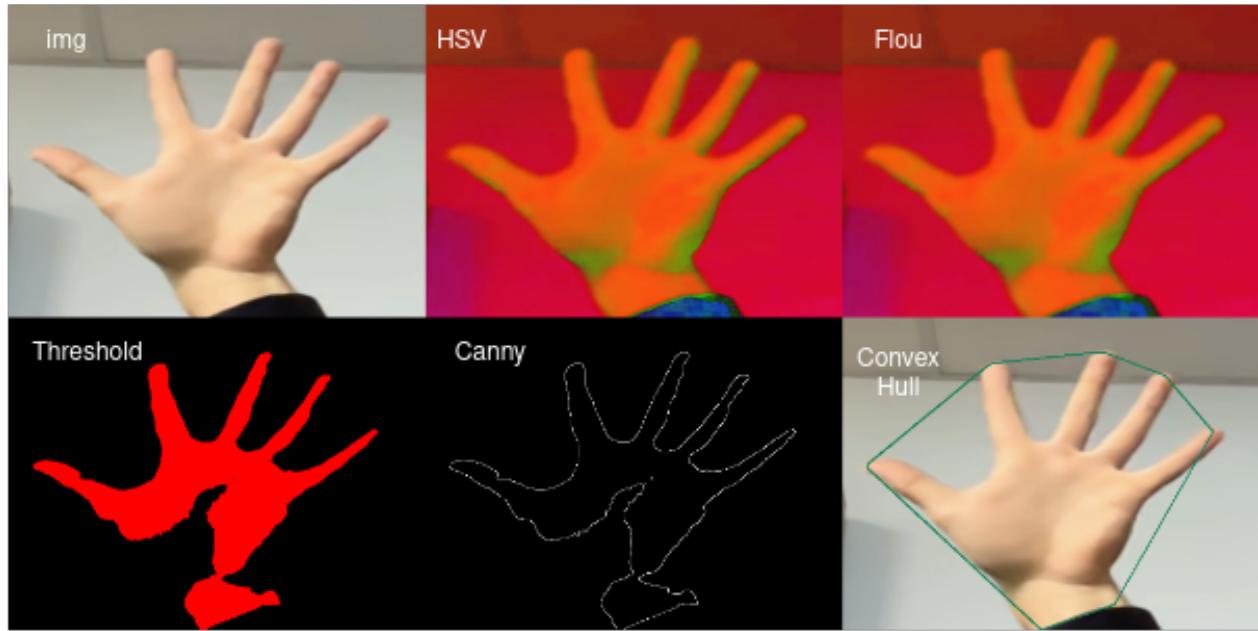


FIGURE 22 – Résultat de la détection de la main : Thresholding : seuil entre 223-255, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

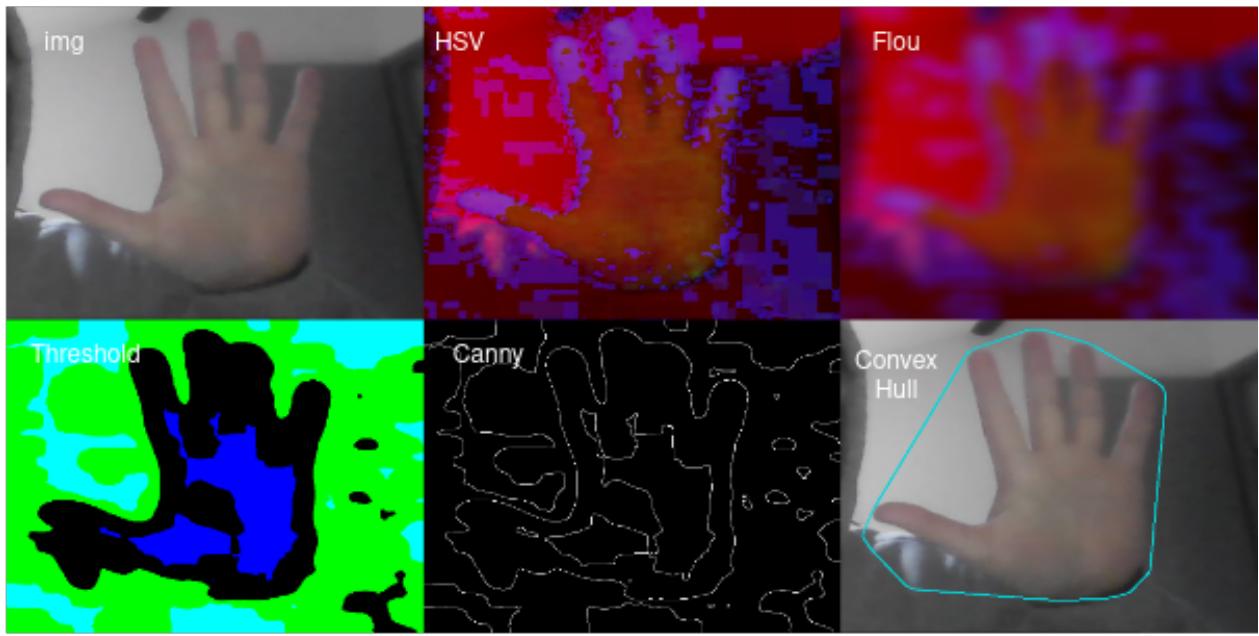


FIGURE 23 – Résultat de la détection de la main : Flou : X = 41, Y = 27, Sigma = 22 ; Thresholding : seuil entre 11-255, TRESH_BINARY ; Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

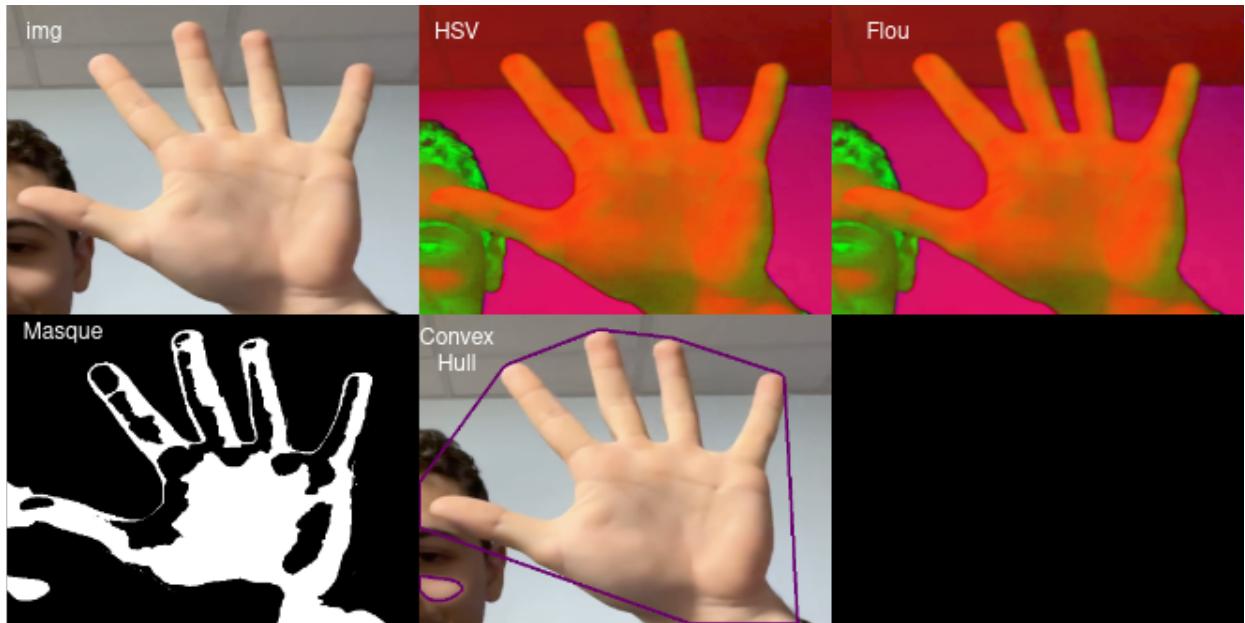


FIGURE 24 – Résultat de la détection de la main : Hue : 0 - 94, Saturation : 37 - 180, Value : 145 - 238

Les résultats obtenus sont bons. Nous arrivons bien à détecter la main. Cependant, ces résultats 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 Convex Hull

Pour la détection des doigts, nous avons créé un masque ayant la forme d'un doigt 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.

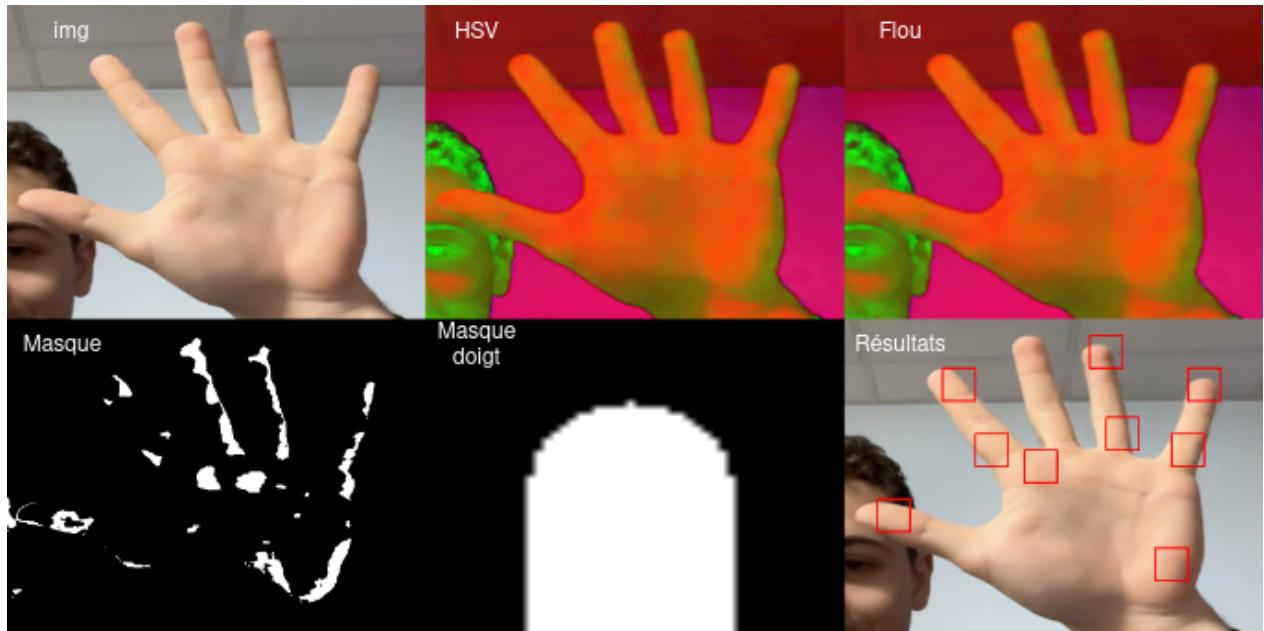


FIGURE 25 – Détection des doigts sans Convex Hull. Hue : 8 - 67; Saturation : 97 - 128; Value : 185 - 239; Seuil : 73 - 100

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 récupérer 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