



Master Informatique

Système de reconnaissance des mouvements de la main

Rapport

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

Étudiants : Victor DALLÉ | Encadrante : Madame Dobrina BOLTCHEVA
Claire KURTH

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

Nous souhaitons exprimer notre gratitude la plus sincère envers Madame Dobrina Boltcheva pour son accompagnement, son soutien et son expertise tout au long de notre projet d'initiation à la recherche. Son engagement, sa patience et ses conseils précieux ont considérablement enrichi notre expérience et ont joué un rôle essentiel dans ce projet. Nous exprimons notre gratitude pour l'occasion qui nous a été offerte de collaborer avec elle, et nous exprimons notre profonde reconnaissance pour son engagement envers notre développement académique. Nous tenons aussi à exprimer notre reconnaissance envers nos familles, nos amis et tous ceux qui ont participé de près ou de loin à la concrétisation de ce projet.

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 et technologies utilisées pour la reconnaissance des gestes de la main	3
2.1 Article de départ	3
2.2 Mediapipe	3
2.3 Classifieur Haar-cascade	4
3 Reconnaissance de la main avec un classifieur Haar-cascade	5
3.1 Entrainer un classifieur Haar-cascade	5
3.2 Nos entraînements	6
3.2.1 Avec une base de données d'images de mains	6
3.2.1.1 Méthodologie	6
3.2.1.2 Résultats	8
3.2.1.3 Conclusion	9
3.2.2 En créant nous même des images positives	9
3.2.2.1 Méthodologie	9
3.2.2.2 Résultats	10
3.2.2.3 Conclusion	11
4 Reconnaissance de la main par traitement d'image	12
4.1 Méthodologie	12
4.2 Expériences et résultats	12
4.3 Conclusion	16
5 Fusion des deux méthodes	16
5.1 Méthodologie	16
5.2 Expériences et résultats	17
5.3 Conclusion	18
6 Détection des mouvements de la main	19
6.1 Sans Convex Hull	19
6.1.1 Méthodologie	19
6.1.2 Expériences et résultats	20
6.1.3 Conclusion	22
6.2 Avec Convex Hull	23

6.2.1	Méthodologie	23
6.2.2	Expériences et résultats	23
6.2.3	Conclusion	28
Conclusion		29
Bibliographie		30

Introduction

De nos jours, la vision par ordinateur représente 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, développé par Google [5] 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 [7] 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 comprendre les mécanismes et les problématiques liés à la reconnaissance des gestes de la main. L'objectif principal est de concevoir un système capable de détecter et de classifier différentes positions précises de la main effectuées par l'utilisateur. Ces positions sont des gestes simples tels que le poing fermé ou alors un certain 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 et le langage de programmation Python en version 3. Nous parlerons dans un premier temps plus en détail des techniques de reconnaissance de gestes actuellement utilisées, puis nous verrons comment nous avons mis en place notre système de reconnaissance de la main. Tout d'abord avec un classifieur Haar-cascade puis par traitement d'images. Nous exposerons ensuite comment nous avons fusionné ces deux méthodes pour obtenir une meilleure détection de la main. Enfin, nous expliciterons comment nous avons détecté les mouvements de la main à l'aide des techniques de reconnaissance de gestes que nous avons mises en place.

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, etc.) et le faire correspondre à différentes tâches comme le lancement d'applications : le bloc-notes, la caméra, l'ouverture de sites web, etc.

L'objectif de ce projet est également de comprendre les mécanismes et les problématiques liés à la reconnaissance des gestes de la main à travers différentes techniques de traitement d'images. Le système doit être mis en œuvre avec l'aide de la librairie de "Computer Vision" - OpenCV, comme dans l'article [7]. Une extension possible serait l'implémentation d'un système de détection des mouvements de la tête, ou du corps tout entier.

1.2 Encadrement

Pendant tout ce semestre, dans le cadre de l'*UE Initiation à la recherche*, nous avons été encadré par Madame Dobrina Boltcheva, enseignante-chercheuse à l'Université de Lorraine. Elle fait partie de l'équipe PIXEL du Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA). C'est une équipe de recherche en traitement numérique de la géométrie et s'intéresse particulièrement aux techniques de paramétrisation, de maillage et de reconstruction d'objets à partir de nuages de points 3D.

2 État de l'art et technologies utilisées pour la reconnaissance des gestes de la main

2.1 Article de départ

L'article sur lequel nous nous basons, *Static Hand Gesture and Face Recognition System* [7] propose un système de reconnaissance de gestes de la main. Ce système est créé à partir d'une image, passée en HSV à laquelle les chercheurs ont ajouté un flou gaussien et un seuillage (thresholding). Le thresholding permet de générer une image binaire : chaque pixel est comparé à un seuil, si la valeur du pixel est supérieure à ce seuil le pixel est blanc, sinon il est noir. Les membres de l'équipe de recherche extraient ensuite les contours avant de tracer le Convex Hull, le polygone entourant la main. Enfin, ils calculent des "Convexity Defect" c'est-à-dire des points éloignés de points convexe. Dans ce cas ci, les points convexes sont le bout des doigts et donc les defects sont l'espace creux entre 2 doigts. Un defect est donc compté si l'angle entre 2 doigts est inférieur à 90°.

2.2 Mediapipe

Mediapipe [5] 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 du corps entier. Il est basé sur des modèles de machine learning notamment grâce à de l'apprentissage via des réseaux profonds de neurones.

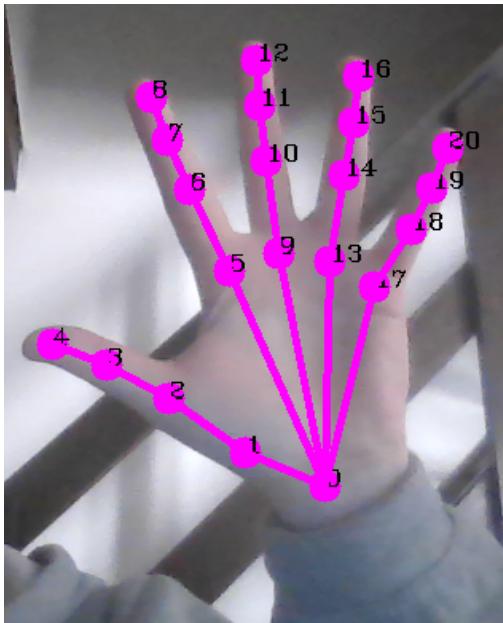


FIGURE 1 – Exemple de détection de la main avec Mediapipe

2.3 Classifieur 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 [10]. 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 plusieurs zones spécifiques de la fenêtre. Ces caractéristiques sont ensuite utilisées pour entraîner un classifieur qui permet de détecter des objets dans une image.



FIGURE 2 – Exemple de détection de visage avec un classifieur Haar-cascade

Les classificateurs Haar-cascade sont utilisés pour la détection de visages (Fig. 2), de voitures, de plaques d'immatriculation, de piétons ou de tout autres objets. Ils sont utilisés dans le domaine de la vision par ordinateur et sont efficaces pour la détection d'objets. Cependant, ils tendent à être remplacés par les réseaux de neurones profonds tels que les réseaux neuronaux convolutifs (CNN) qui sont plus performants pour la détection d'objets.

3 Reconnaissance de la main avec un classifieur Haar-cascade

3.1 Entrainer un classifieur Haar-cascade

Pour entraîner un classifieur Haar-cascade [8], il faut tout d'abord collecter des images positives et négatives. Les images positives sont des images contenant l'objet que nous souhaitons détecter, tandis que les images négatives sont des images ne contenant pas l'objet. Il faut ensuite générer des fichiers de description 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 description.

L'entraînement du classifieur en lui-même se fait grâce à des "features". Ces dernières ont été introduites par Viola et Jones en 2001 (Fig. 3). Par la suite, d'autres features ont été ajoutées (Fig. 4) afin d'améliorer la détection d'objets dans une image.

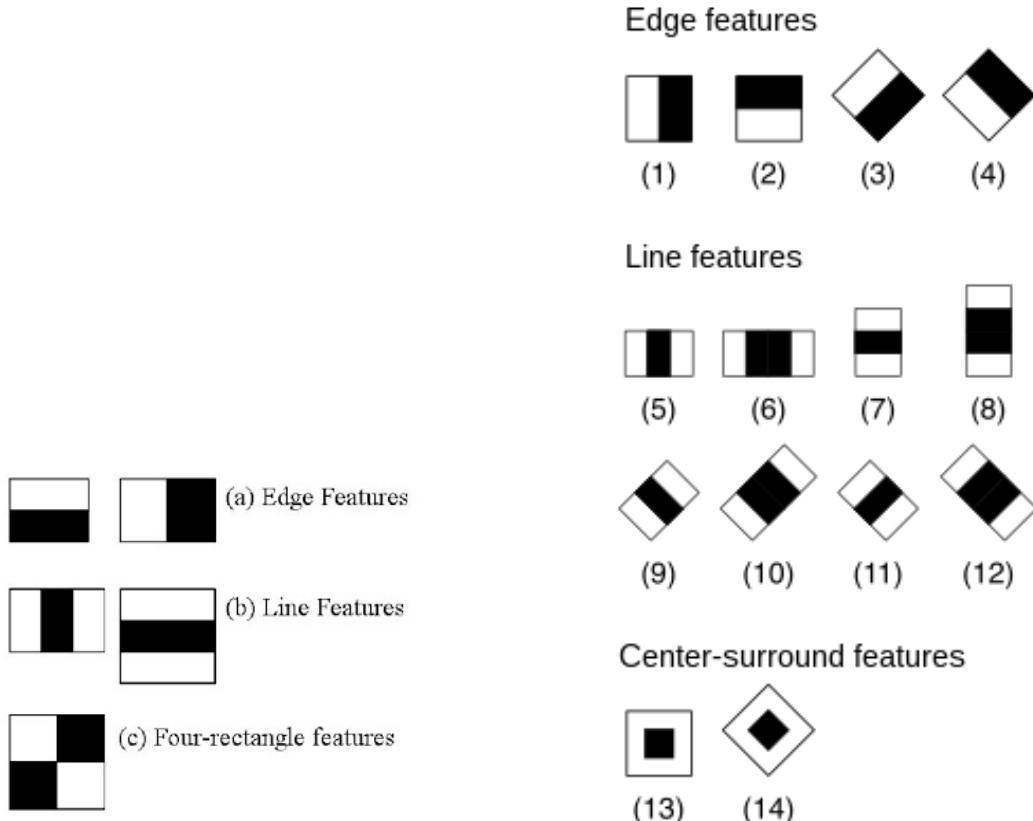


FIGURE 3 – Features de Haar comme utilisées par Viola et Jones [9]

FIGURE 4 – Features de Haar supplémentaires [4]

Ces features sont des caractéristiques de l'objet que nous souhaitons 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 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 [1] 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.

3.2 Nos entraînements

3.2.1 Avec une base de données d'images de mains

3.2.1.1 Méthodologie

Pour notre entraînement, nous avons collecté 10 000 images négatives (Fig. 5) [6, 3] et 5 000 images positives de mains (Fig. 6) [2]. Nous avons dû pour les images positives, les annoter (dans un fichier texte) afin de donner le nombre de mains présentes et les coordonnées de celles-ci dans l'image (Fig. 7). Étant donné que seule une main était présente et que le fond est blanc, la zone où est située la main est donc l'image entière. 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 fichier.



FIGURE 5 – Exemple d'image négative



FIGURE 6 – Exemple 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 7 – Exemple de fichier de description

Nous sommes ensuite passés à l'entraînement. Nous avons utilisé pour cela OpenCV [9] 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 : 'Default' et 'All'. Le mode 'Default' utilise les features de bases (Fig. 3) tandis que le mode 'All' utilise les features un peu plus complexes (Fig. 4). 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	Default	1h10
2	4 000	10 000	8	1	1.0e-5	Default	40min
3	4 000	2 000	10	3	1.0e-5	All	5h
4	4 000	2 000	6	1	1.0e-5	Default	4mn

3.2.1.2 Résultats

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

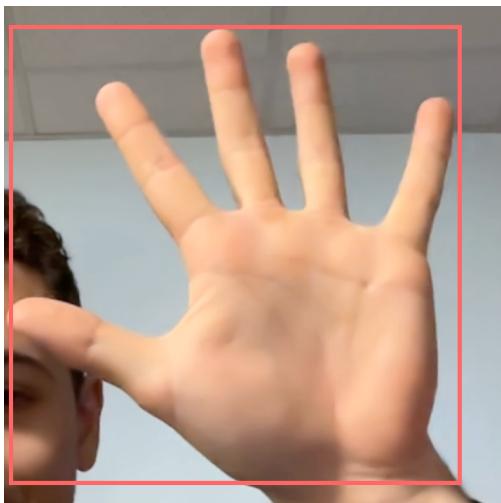


FIGURE 8 – Résultat attendu

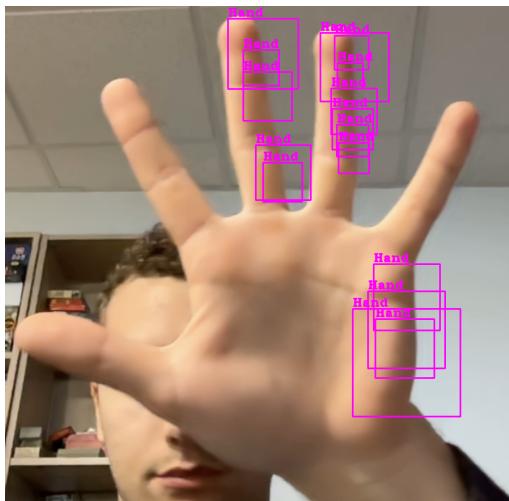


FIGURE 9 – Test 1

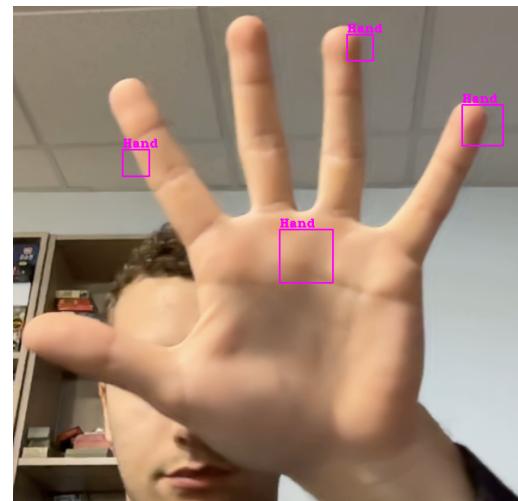


FIGURE 10 – Test 2



FIGURE 11 – Test 3

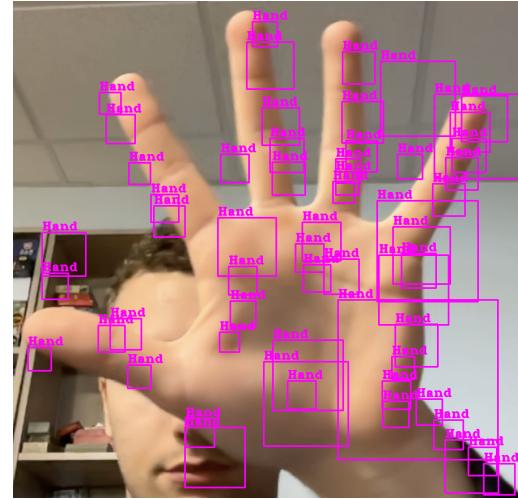


FIGURE 12 – Test 4

Les classifieurs 1 (Fig. 9) et 2 (Fig. 10) ne détectent pas entièrement la main dans l'image, seulement plusieurs portions plus ou moins grandes. Le classifieur 4 (Fig. 12) détecte bien la main mais détecte aussi d'autres objets dans l'image. Le classifieur 3 (Fig. 11) est le moins satisfaisant puisqu'il ne détecte rien du tout, certainement dû à un surapprentissage.

3.2.1.3 Conclusion

Les résultats ne sont pas entièrement satisfaisants. En effet, les différents classifieurs, lorsqu'ils détectent la main, ne la détectent pas entièrement mais seulement plusieurs parties. Cela 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

3.2.2.1 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 négatives (Fig. 13). Nous avons ensuite généré les fichiers de description des images positives et entraîné le classifieur à l'aide de ces fichiers.



FIGURE 13 – Exemple d'image positive créée à partir d'une image négative

Comme précédemment, nous avons testé 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	Default	4h05
6	8 000	6 000	10	1	1.0e-5	Default	1h53
7	8 000	6 000	5	1	1.0e-5	Default	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

3.2.2.2 Résultats



FIGURE 14 – Test 5

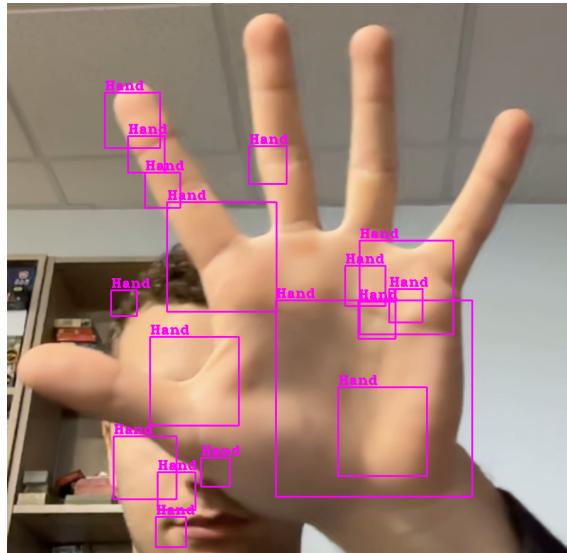


FIGURE 15 – Test 6

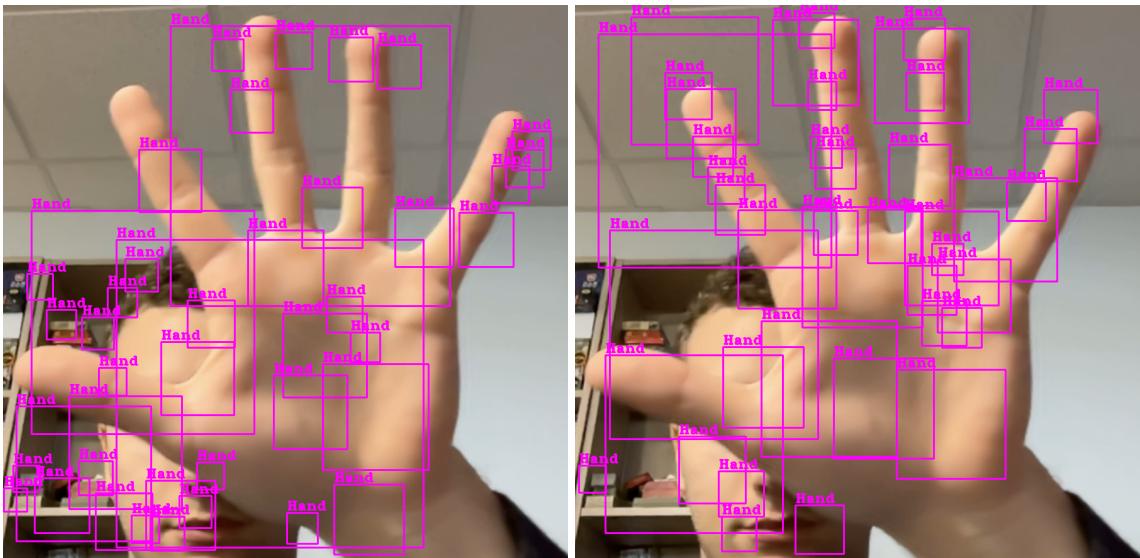


FIGURE 16 – Test 7

FIGURE 17 – Test 8

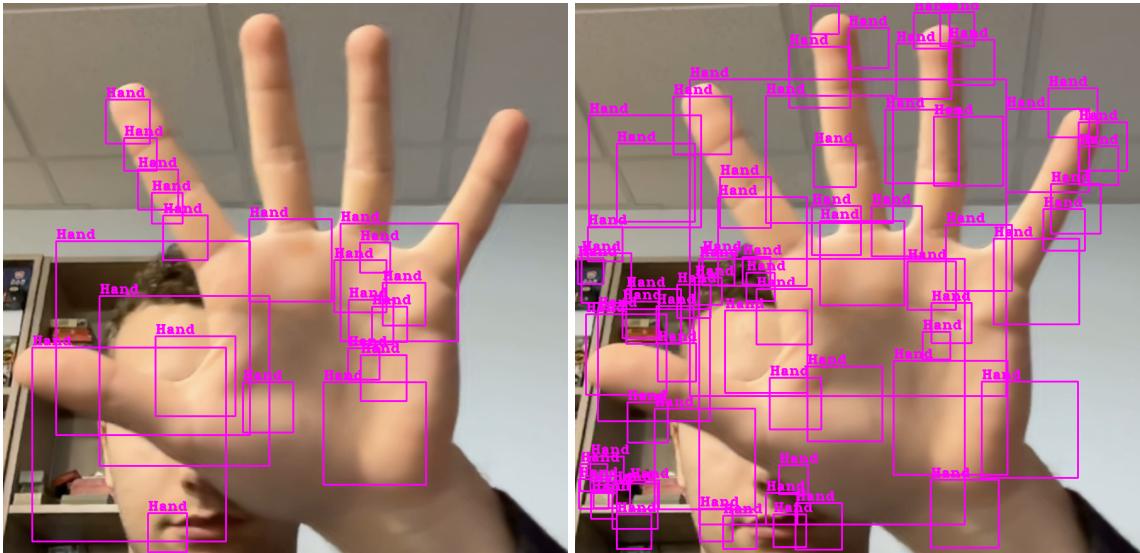


FIGURE 18 – Test 9

FIGURE 19 – Test 10

Les résultats ne sont pas satisfaisants. Le classifieur 5 (Fig. 14) ne détecte rien du tout, certainement dû à un surapprentissage. Les classificateurs 7 (Fig. 16), 8 (Fig. 17) et 10 (Fig. 19) détectent bien la main mais détectent aussi beaucoup d'autres objets dans l'image. Les classificateurs 6 (Fig. 15) et 9 (Fig. 18) 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.

3.2.2.3 Conclusion

Les résultats ne sont pas beaucoup plus satisfaisant. Nous obtenons dans l'ensemble, les mêmes résultats que précédemment c'est-à-dire des détections partielles de la main et non la main entière pour les classificateurs les mieux entraînés.

Une piste d'amélioration serait d'obtenir une base de données contenant des images de mains avec des fonds et des positions différentes. Il faudrait également annoter manuellement les images positives pour sélectionner les coordonnées de la main dans l'image. Cela permettrait au classifieur de mieux apprendre à détecter la main dans une image et ainsi d'obtenir de meilleurs résultats.

4 Reconnaissance de la main par traitement d'image

4.1 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. Nous avons 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.

4.2 Expériences et résultats

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

Tout d'abord, nous avons essayé de transformer l'image en niveaux de 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 tracé 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.

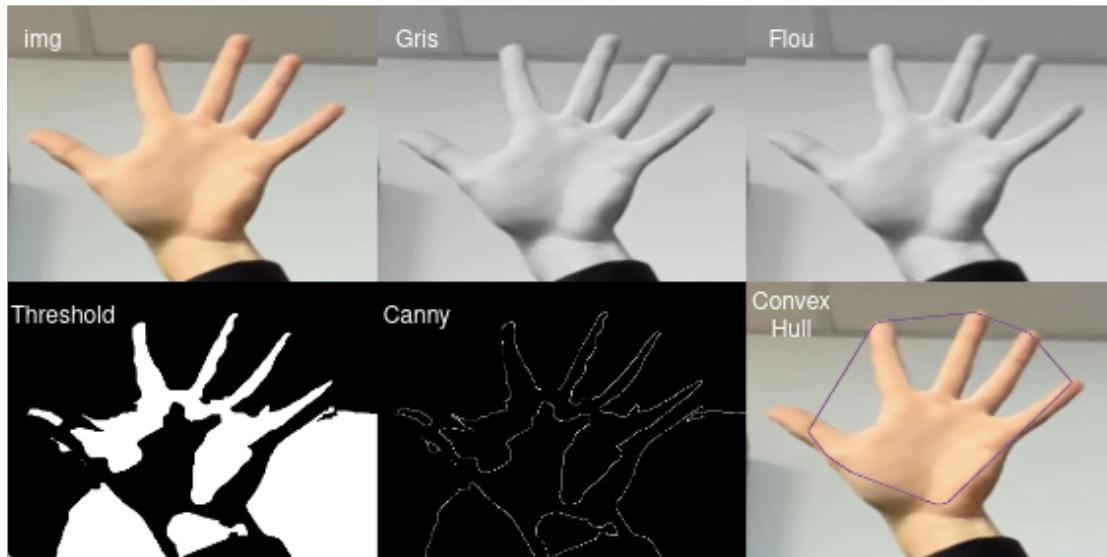


FIGURE 20 – Résultat de la détection de la main : Thresholding : seuil entre 200-255, TRESH_BINARY,
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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

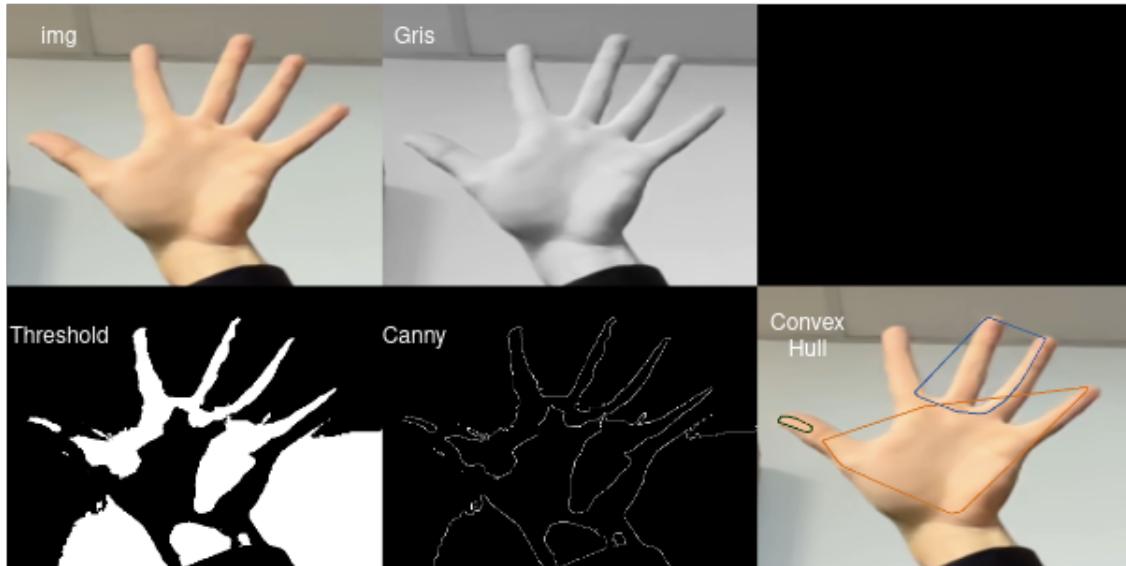


FIGURE 21 – Résultat de la détection de la main : Thresholding : seuil entre 198-255, TRESH_BINARY,
Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

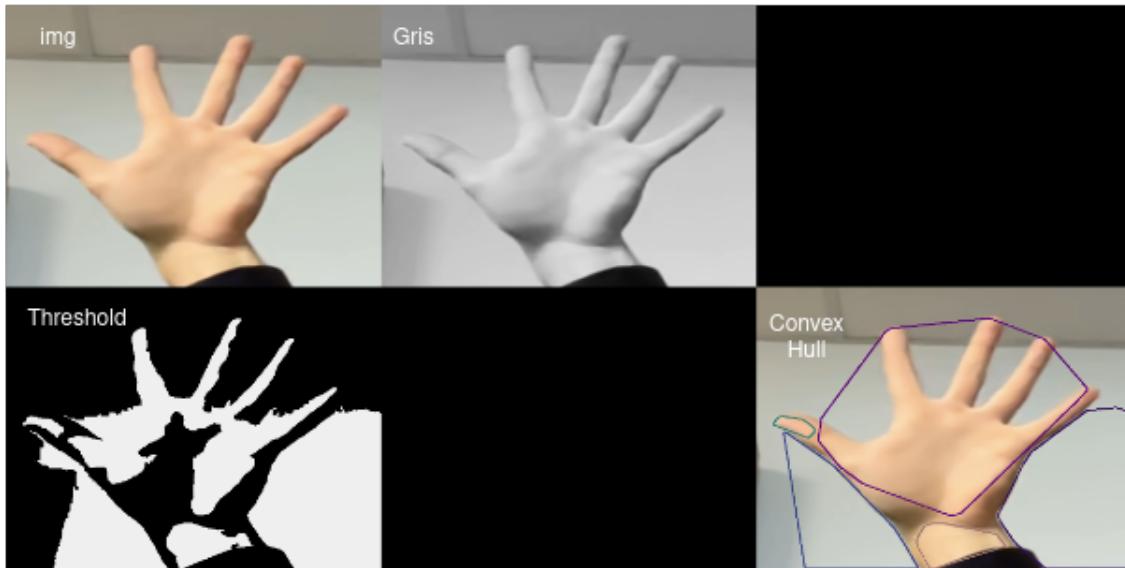


FIGURE 22 – Résultat de la détection de la main : Thresholding : seuil entre 196-239, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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.

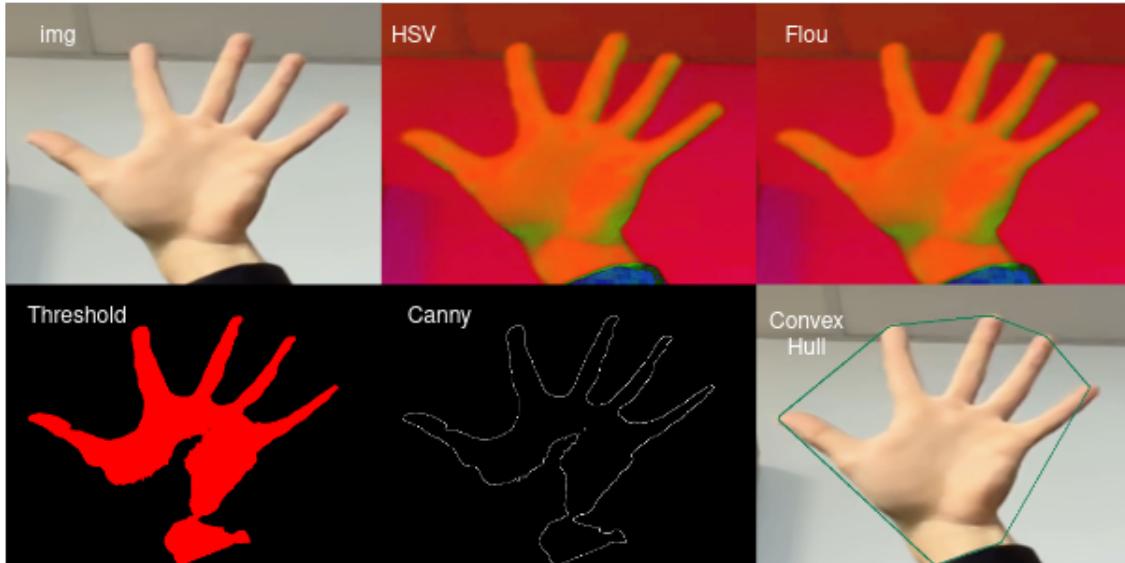


FIGURE 23 – Résultat de la détection de la main : Thresholding : seuil entre 223-255, TRESH_BINARY, Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

Nous pouvons clairement voir 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.

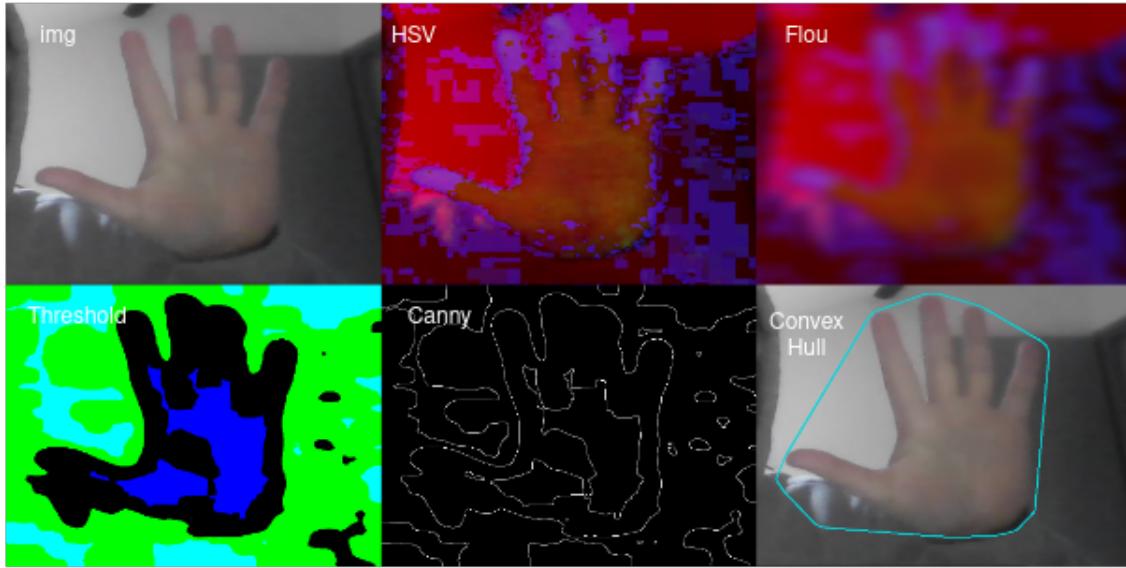


FIGURE 24 – Résultat de la détection de la main : Flou : X = 41, Y = 27, Sigma = 22 ; Thresholding : seuil entre 11-255, TRESH_BINARY ; Contours : RETR_EXTERNAL, CHAIN_APPROX_SIMPLE

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

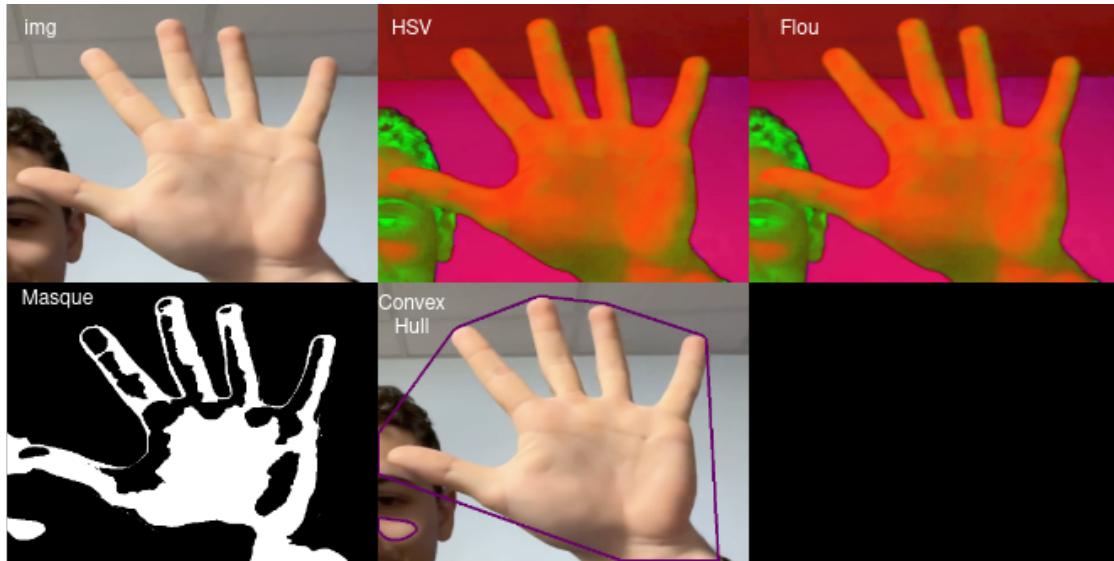


FIGURE 25 – Résultat de la détection de la main : Hue : 0 - 94, Saturation : 37 - 180, Value : 145 - 238

4.3 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. Cependant, l'objectif qui était de détecter la main sans classifieur est atteint car dans la plupart des cas, nous avons bien détecté la main.

5 Fusion des deux méthodes

5.1 Méthodologie

Nous avons un classifieur d'une part qui détecte une portion de la main et d'autre part un système qui permet de détecter la main si les paramètres sont bien configurés. Nous allons donc dans cette partie, essayer de fusionner les deux méthodes pour obtenir une détection automatique de la main sans avoir à régler des paramètres.

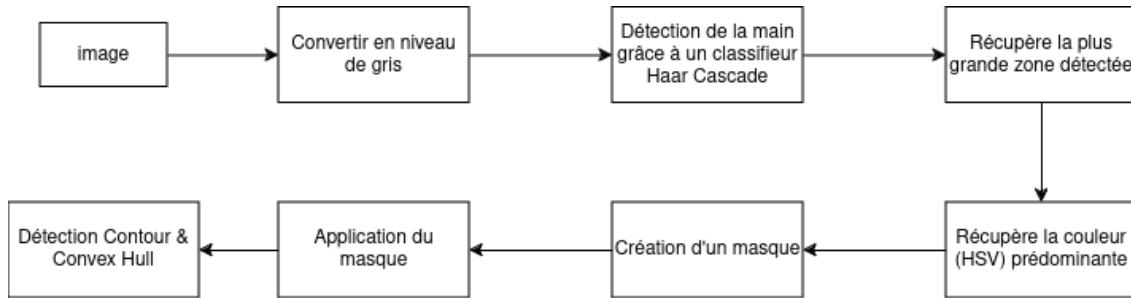


FIGURE 26 – Pipeline de la fusion des deux méthodes

La méthodologie est la suivante (Fig. 26) : à partir d'un image que nous convertissons en niveau de gris, nous essayons de détecter la main grâce à un classifieur Haar-cascade entraîné. Nos classificateurs détectant plusieurs zones au niveau de la main, nous récupérons la plus grosse. Nous passons ensuite cette zone en HSV afin de récupérer la teinte, saturation et la valeur majoritaire de la zone. Nous créons ensuite un masque en fonction de ces valeurs et d'un epsilon afin d'avoir un seuil minimum et un seuil maximum. Nous passons ensuite ce masque sur l'image et affichons le Convex Hull correspondant.

5.2 Expériences et résultats

Nous pouvons voir (Fig. 27) que la fusion des deux méthodes fonctionne bien. Nous avons bien détecté la main et les 5 doigts. Cependant, cette méthode ne semble fonctionner qu'avec une main ouverte et 5 doigts levés.

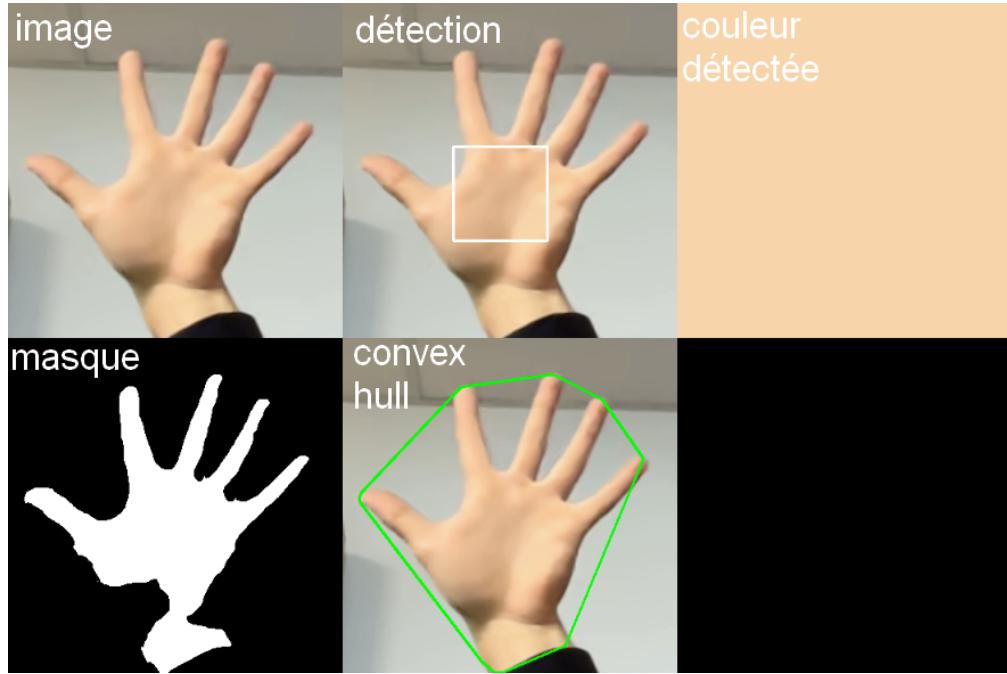


FIGURE 27 – Résultat de la fusion des deux méthodes.

En effet, lorsque nous avons ensuite essayé avec 2 doigts levés (Fig. 28), une partie de la main est bien détecté. Cependant, étant donné que nous récupérons la couleur majoritaire de la zone détecté, nous avons ici la couleur du fond de l'image et non de la main car même si la zone représentant une partie de la main semble être légèrement plus grande, les nuances de couleurs y sont aussi beaucoup plus nombreuses.

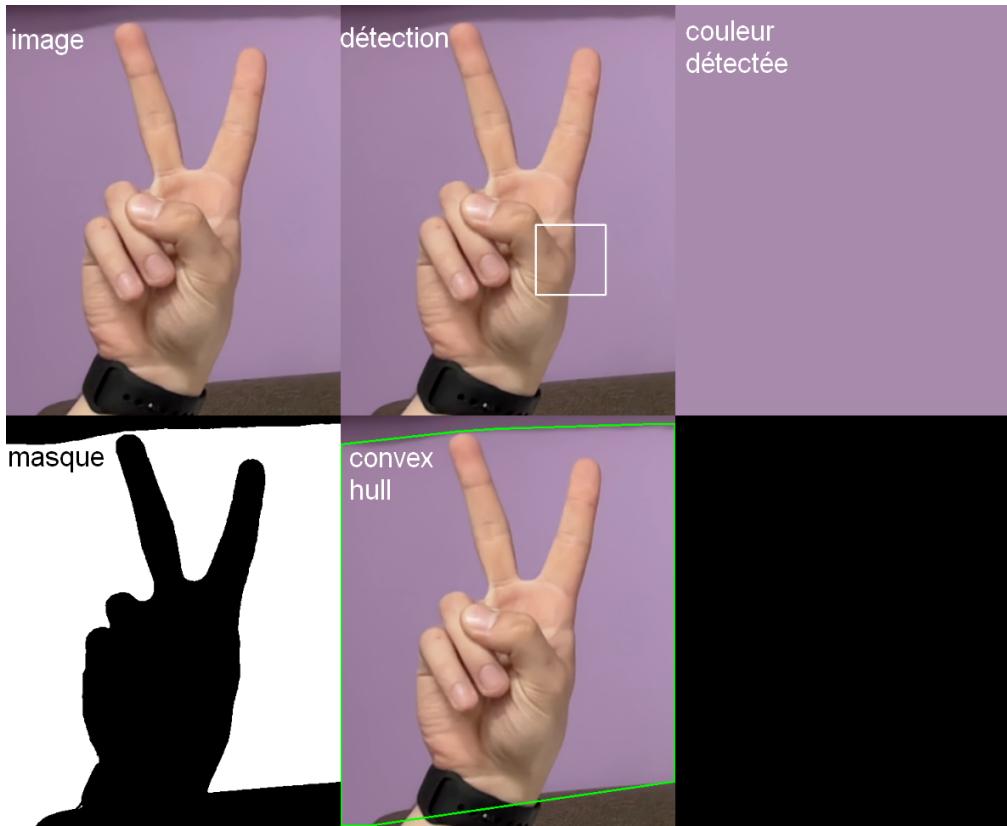


FIGURE 28 – Résultat de la fusion des deux méthodes avec 2 doigts levés.

5.3 Conclusion

Cette méthode rencontre deux problèmes principaux. Tout d'abord, elle fonctionne beaucoup mieux sur une main ouverte avec 5 doigts levés car le classifieur a été entraîné pour reconnaître une main ouverte. De plus, récupérer la couleur majoritaire de la zone détectée ne semble pas être la meilleure méthode pour récupérer la couleur de la main. En effet, comme nous avons pu le voir précédemment, les variations dues à la luminosité ou à la qualité de l'image peuvent faire en sorte que la couleur majoritaire ne soit pas la couleur de la main. Enfin, dans le but d'avoir une meilleure détection, nous utilisons un paramètre "epsilon" qui permet d'avoir un intervalle de valeurs plus ou moins grand pour la teinte, la saturation et la valeur. Cependant, ce paramètre peut être problématique car il dépend de la qualité de l'image.

6 Détection des mouvements de la main

Étant donné que la fusion des deux méthodes ne fonctionne pas pour tous les cas, nous avons décidé de repartir sur une méthode où nous pouvons modifier les valeurs des paramètres pour la détection de la main.

6.1 Sans Convex Hull

6.1.1 Méthodologie

Nous avons une image que nous passons en HSV à laquelle nous ajoutons un flou Gaussien. Avec un système de trackbars nous ajustons ensuite les valeurs minimales et maximales pour la teinte, la luminosité et la valeur afin d'avoir un masque qui ne détecte que la main. Nous passons ensuite ce masque sur l'image et détectons les contours de la main. Pour la détection des doigts, nous avons ensuite créé différents masques 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.

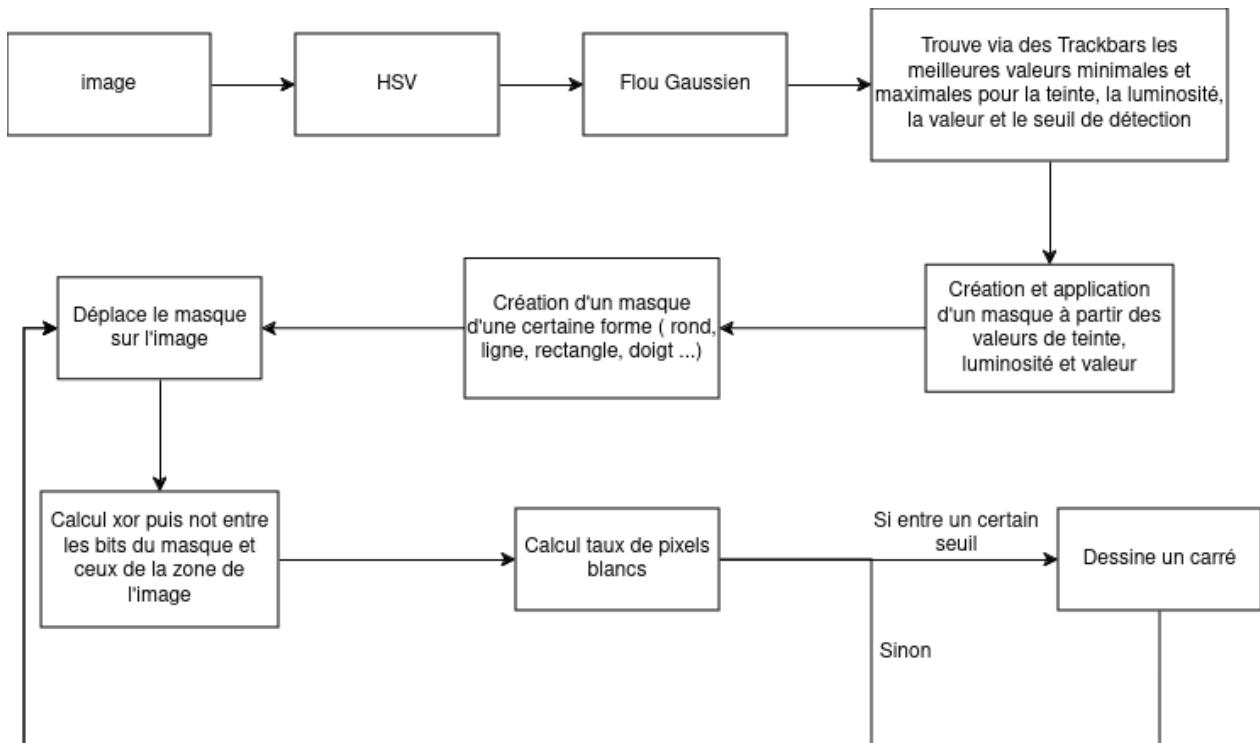


FIGURE 29 – Pipeline de la détection des doigts sans Convex Hull

6.1.2 Expériences et résultats

Dans cet exemple (Fig. 30), nous avons réalisé un masque de type circulaire pour faire correspondre le bout des doigts avec le masque. Le résultat n'est pas probant car les zones détectées ne correspondent pas aux doigts.

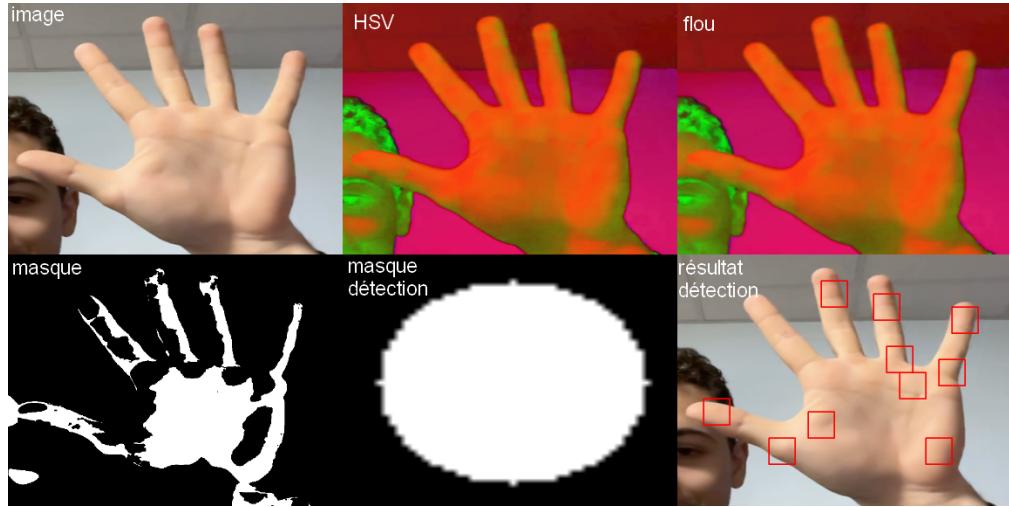


FIGURE 30 – Détection des doigts sans Convex Hull. Teinte : 10 - 43 ; Saturation : 79 - 150 ; Valeur : 174 - 237 ; Seuil : 65 - 75

Ici, le masque a été modifié pour être de forme rectangulaire. Le résultat (Fig. 31) est meilleur que précédemment mais il reste des zones non désirées. On remarque que les zones détectées correspondent pour certaines au bout des doigts comme souhaité.

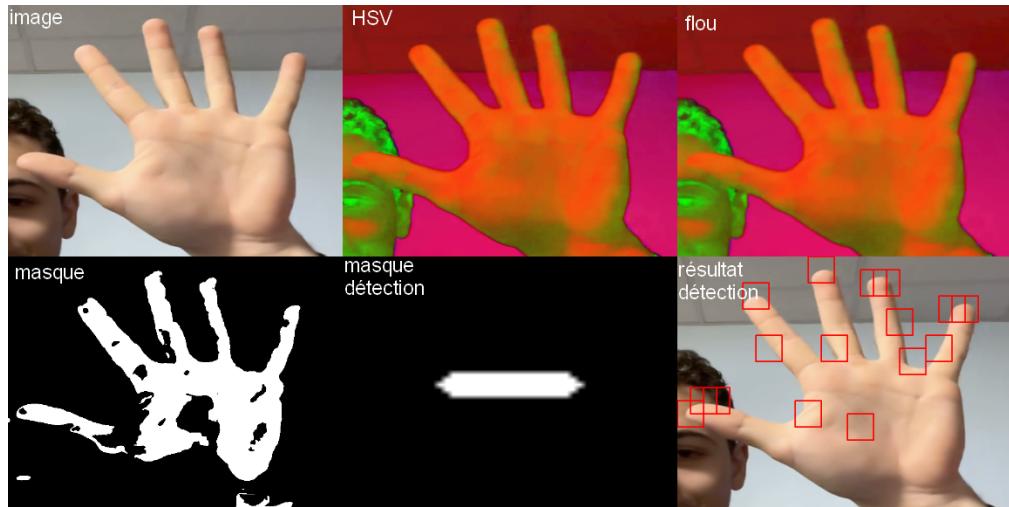


FIGURE 31 – Détection des doigts sans Convex Hull. Teinte : 9 - 41 ; Saturation : 79 - 150 ; Valeur : 202 - 255 ; Seuil : 62 - 75

Dans cette tentative (Fig. 32), le masque a été de nouveau modifié pour être épouser la forme d'un doigt avec une partie rectangulaire et arrondie au sommet. On retrouve un peu moins d'anomalies que précédemment. Cependant, il reste des zones non désirées ainsi que des doigts non détectés (le majeur).

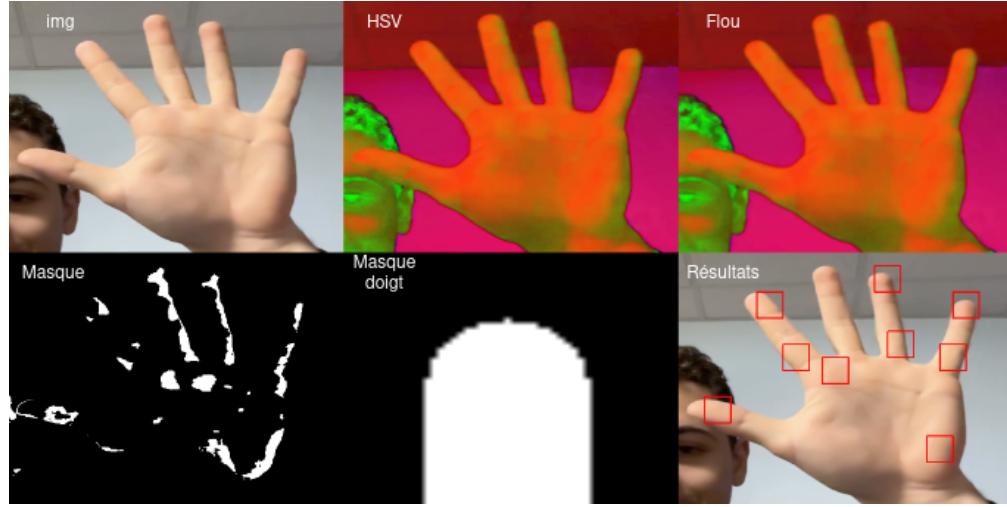


FIGURE 32 – Détection des doigts sans Convex Hull. Teinte : 8 - 67 ; Saturation : 97 - 128 ; Valeur : 185 - 239 ; Seuil : 73 - 100

Dans cette avant-dernière tentative (Fig. 33), le masque a été modifié pour être de forme rectangulaire et plus petit. Le résultat est meilleur que précédemment mais il reste des zones non désirées. On peut noter que cette fois, on peut retrouver au moins un carré de détection par doigt.

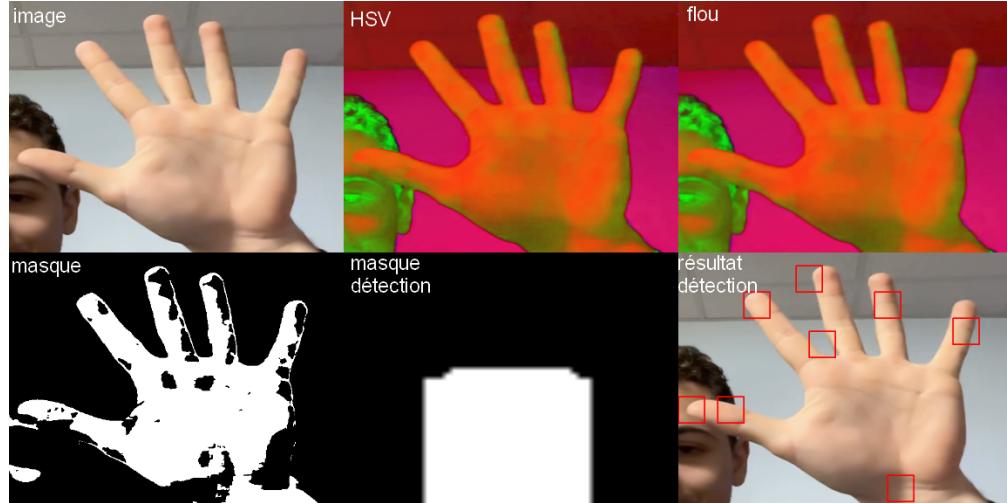


FIGURE 33 – Détection des doigts sans Convex Hull. Teinte : 8 - 59 ; Saturation : 66 - 98 ; Valeur : 112 - 255 ; Seuil : 54 - 57

Dans ce dernier exemple (Fig. 34), le masque contient toujours une partie rectangulaire mais la partie représentant le bout du doigt a été fortement agrandie. Le résultat présenté est le meilleur en terme de précision de détection des doigts. Cependant, on note que le pouce n'est pas détecté et que les 2 derniers doigts n'ont pas vu leur sommet détecté mais seulement une zone vers le milieu du doigt. On remarque aussi une détection supplémentaire au niveau de la paume de la main.

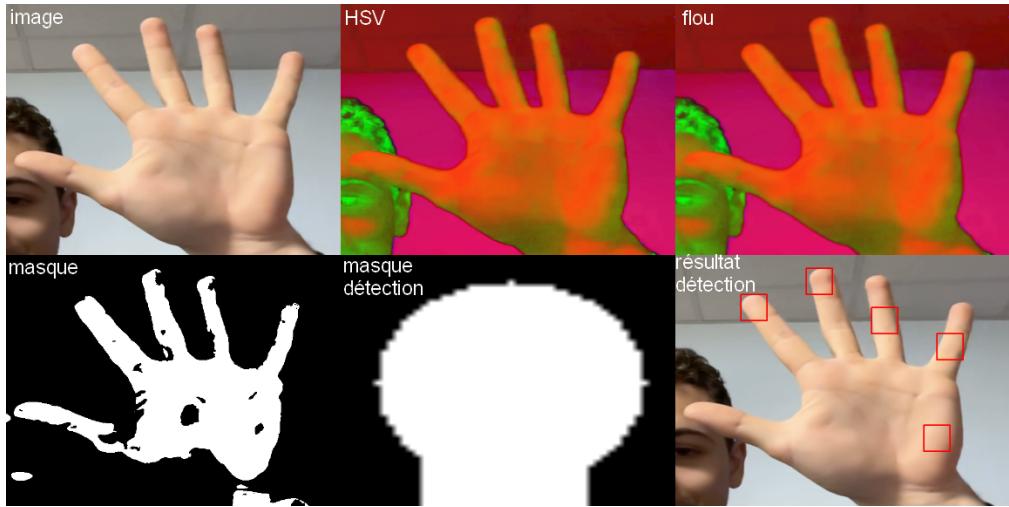


FIGURE 34 – Détection des doigts sans Convex Hull. Teinte : 10 - 72 ; Saturation : 75 - 147 ; Valeur : 193 - 255 ; Seuil : 71 - 100

6.1.3 Conclusion

La détection des doigts sans Convex Hull est une méthode perfectible. En effet, les résultats obtenus ne sont pas tout à fait satisfaisants. Les zones détectées ne correspondent pas toujours aux doigts. Il reste des zones non désirées et des doigts non détectés. Il faudrait donc trouver une méthode pour que le programme puisse trouver les meilleurs paramètres pour la détection des doigts. Aussi, le programme est toujours dépendant de la détection préalable de la main. Si la main n'est pas détectée, la détection des doigts ne pourra pas être effectuée. Cela passe par le réglage des paramètres de filtre HSV qui eux sont manuels.

6.2 Avec Convex Hull

6.2.1 Méthodologie

Le début est le même que précédemment. Nous avons une image que nous passons en HSV à laquelle nous ajoutons un flou Gaussien. Avec un système de trackbars nous ajustons ensuite les valeurs minimales et maximales pour la teinte, la luminosité et la valeur afin d'avoir un masque qui ne détecte que la main. Nous passons ensuite ce masque sur l'image et détectons les contours de la main. Nous utilisons ensuite la fonction Convex Hull d'OpenCV pour tracer le contour de la main. Nous récupérons ensuite les points du Convex Hull et les affichons sur l'image. Nous créons ensuite des filtres afin de n'avoir seulement les points qui nous intéressent.

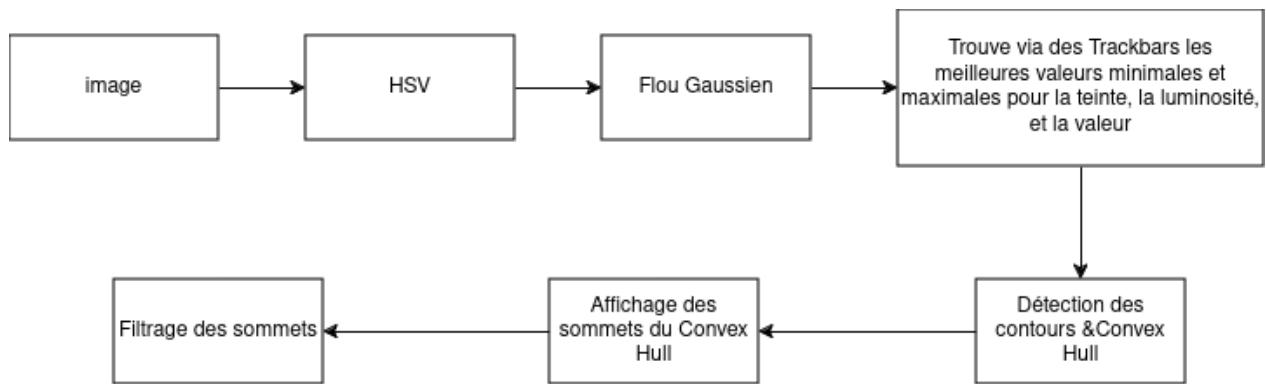


FIGURE 35 – Pipeline de la détection des doigts avec Convex Hull

6.2.2 Expériences et résultats

Comme nous pouvons le voir (Fig. 36), la détection des doigts est plutôt bonne. Nous avons bien des petits cercles qui apparaissent au bout des doigts. Cependant, étant donné que nous affichons les sommets du Convex Hull, nous avons plusieurs points au bout de chaque doigt ainsi qu'en bas de la main.

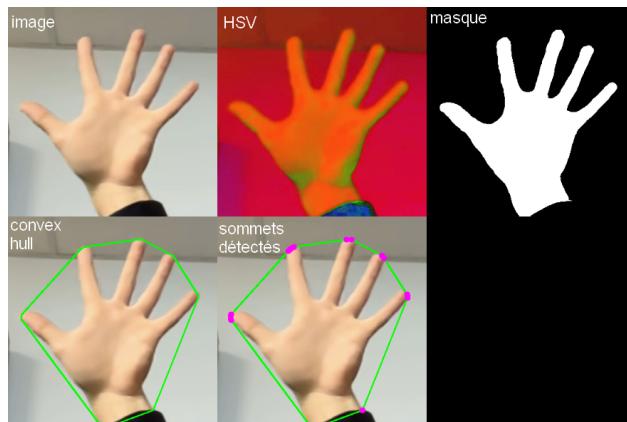


FIGURE 36 – Résultat obtenu avec Convex Hull

Nous avons donc essayé de filtrer les points : tout d'abord au niveau des doigts nous gardons un seul point par doigt. Ensuite, nous filtrons les points en bas de la main : si la coordonnée y d'un point est 1.5 fois inférieure à la moyenne des coordonnées y de tous les points, alors nous le supprimons.

Les résultats sont meilleurs (Fig. 37). Nous avons bien un seul point par doigt et les points en bas de la main ont été supprimés.

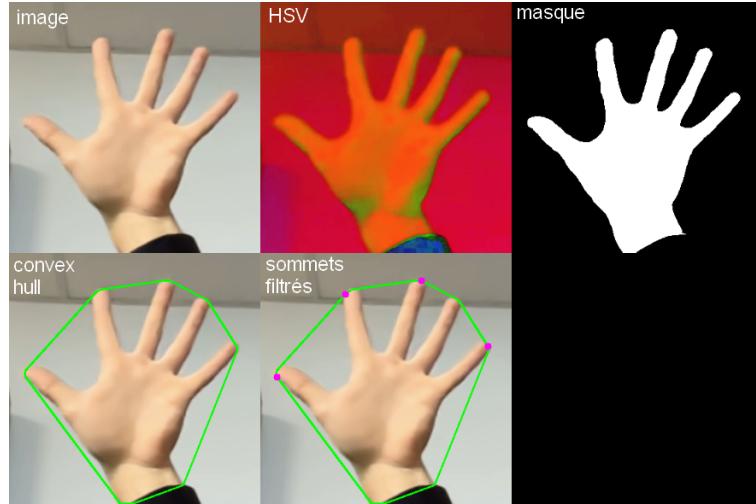


FIGURE 37 – Résultat obtenu avec Convex Hull et filtrage des points

Nous avons donc ensuite essayé sur d'autres images avec moins de doigts levés.

Les résultats (Fig. 38) sont bons avec 4 doigts, nous avons bien seulement 4 points au bout de chaque doigts.

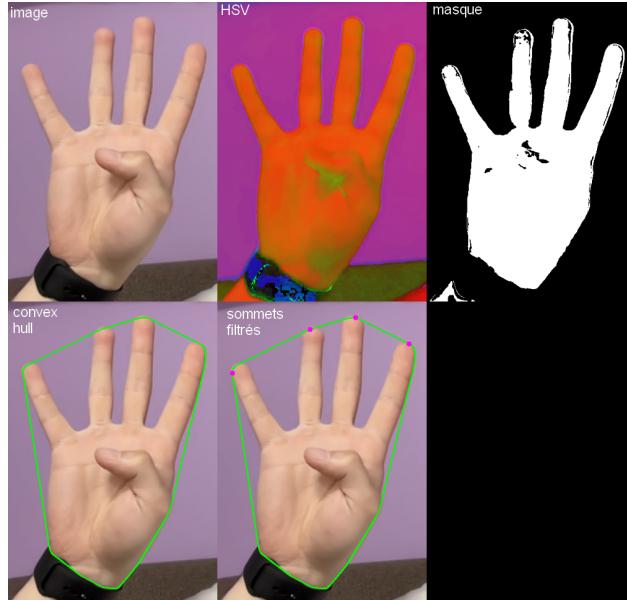


FIGURE 38 – Résultat obtenu avec Convex Hull et filtrage des points

Nous avons testé ensuite, notre pipeline sur une image avec 3 doigts levés. Comme nous pouvons le voir nous avons 2 points à gauche qui ne sont pas des doigts et qui n'ont pas été filtrés.

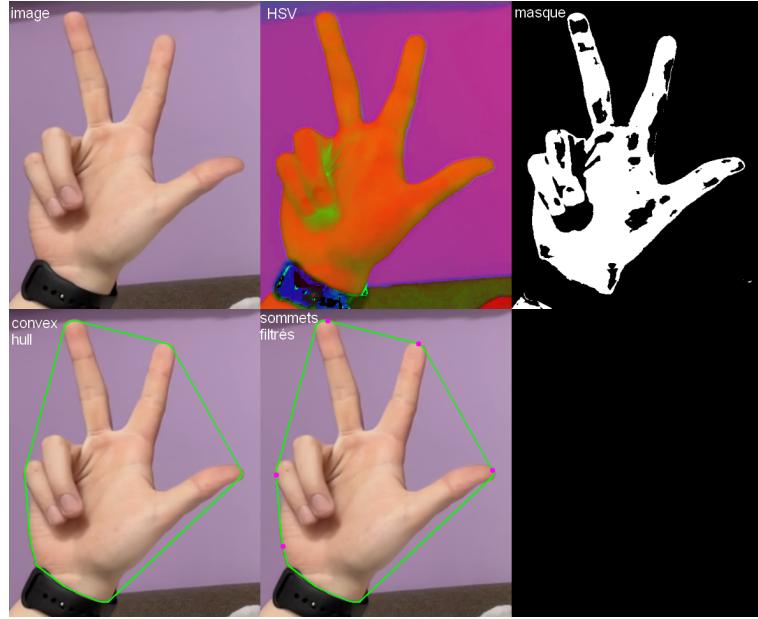


FIGURE 39 – Résultat obtenu avec Convex Hull et filtre des points

Étant donné que ces points étaient sur un même axe vertical nous avons décidé de les filtrer en fonction de leur coordonnées x : si la coordonnée x est sensiblement identique pour deux points, alors nous les supprimons. Les résultats (Fig. 40) sont meilleurs avec cette nouvelle approche. Nous avons bien seulement 3 points au bout de chaque doigts.

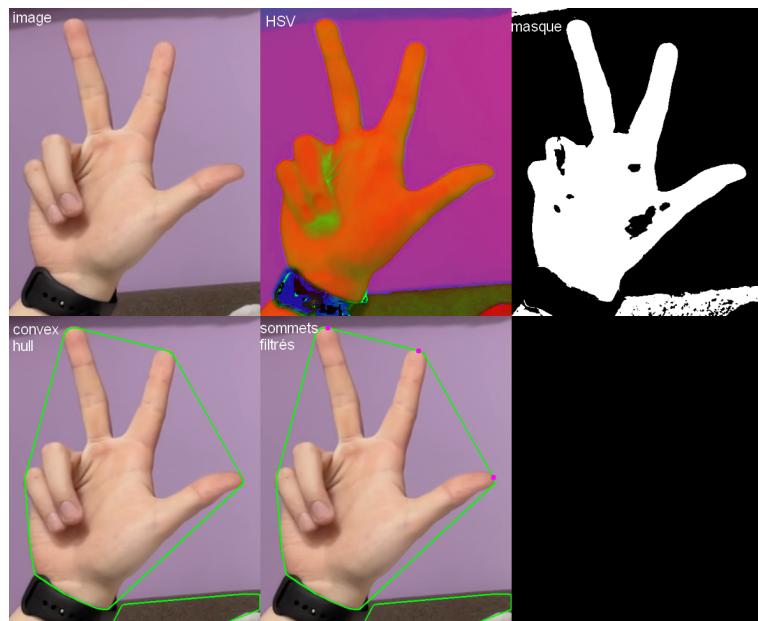


FIGURE 40 – Résultat obtenu avec Convex Hull et filtre des points

Nous avons ensuite testé sur une image avec 2 doigts levés. Les résultats sont cependant moins bons que prévu (Fig. 41). Nous avons seulement un seul point au lieu de deux. Lorsque nous avons regardé de plus près le problème, nous nous sommes rendus compte que le point au niveau de l'index était supprimé car un autre point était présent sur le même axe vertical (Fig. 42).

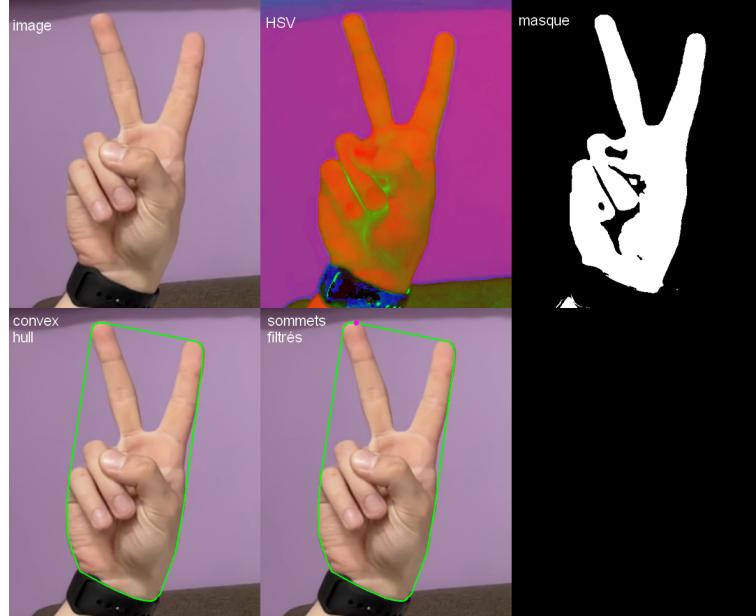


FIGURE 41 – Résultat obtenu avec Convex Hull et filtrage des points

Le point au niveau de l'index est donc supprimé car il y en a deux autres sur le même axe vertical. Nous avons donc décidé de filtrer autrement : au lieu de supprimer tous les points qui sont sur un (quasi) même axe, nous avons décidé de pondérer les points en fonction de leur coordonnées en y. Le point le plus haut aura un poids de 1, le point le plus bas un poids de 0. Pour tous les points qui seront sur un même axe vertical, nous regardons lequel est le plus grand et si le poids du plus grand est supérieur à 0.8 (donc s'il est proche du haut), alors nous le gardons.

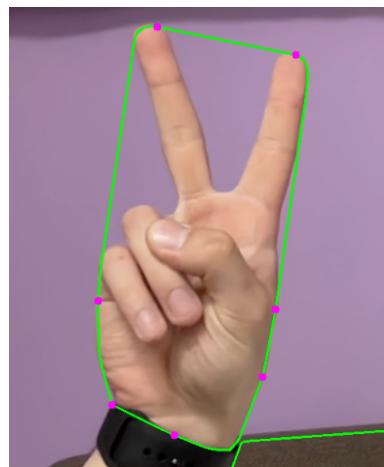


FIGURE 42 – Résultat obtenu avec Convex Hull et tous les points des sommets

En combinant les 2 approches, nous avons décelé un autre problème avec le filtrage en y. Seuls les points de la base de la main sont affichés.

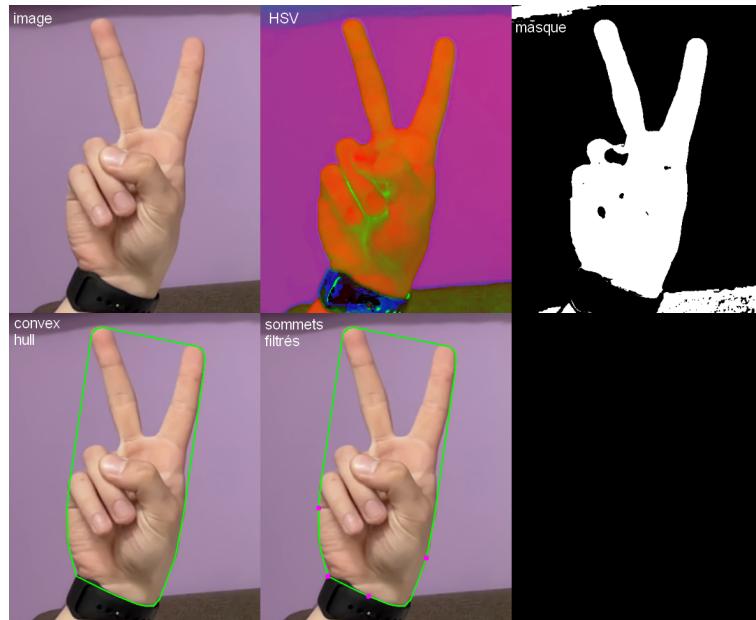


FIGURE 43 – Résultat obtenu avec Convex Hull et filtrage des points

Pour palier ce problème, au lieu de garder les points 1.5 fois plus petits que la moyenne, nous avons décidé de garder les points 0.5 fois plus petits que la moyenne et les résultats sont meilleurs (Fig. 44). Nous avons bien deux points au bout de chaque doigts.

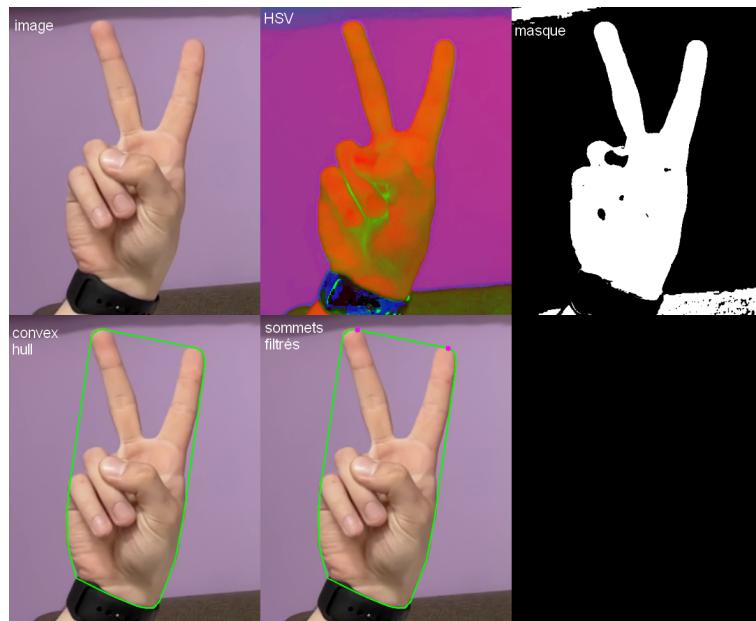


FIGURE 44 – Résultat obtenu avec Convex Hull et filtrage des points

Nous avons donc re-testé avec ces nouveaux filtres sur plusieurs images avec 5, 4 et 3 doigts (Fig. 45). Les résultats ne sont pas aussi bons. En effet, nous pouvons voir qu'il n'y a que deux points sur l'image de 5 doigts et deux sur l'image de 3 doigts.

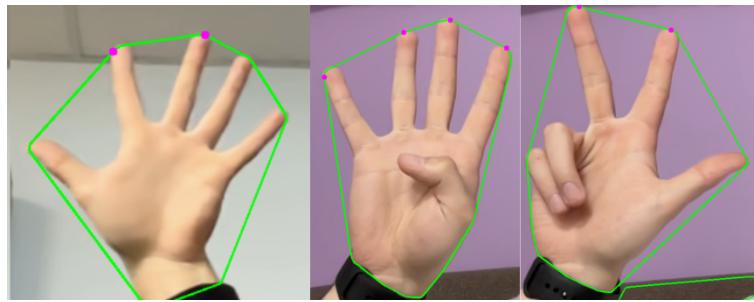


FIGURE 45 – Résultat obtenu avec Convex Hull et filtrage des points

6.2.3 Conclusion

Les résultats obtenus sont plutôt encourageants, nous avons réussi dans certains cas à détecter le bon nombre de doigts. Cependant, les résultats ne sont pas parfaits. En effet, les filtres que nous avons mis en place ne fonctionnent pas pour tous les cas : quand l'un est adapté pour un cas, il ne l'est pas pour un autre. L'enjeu suivant serait de trouver une autre approche unificatrice pour les filtres afin d'avoir une détection des doigts plus précise et plus fiable et qui fonctionne pour tous les cas.

Conclusion

En synthèse, notre exploration des méthodes de détection des mains et des doigts a révélé des résultats encourageants, bien que perfectibles. Malgré la configuration minutieuse des paramètres, la détection de la main a toutefois donné des résultats satisfaisants. Cependant, la reconnaissance des doigts se révèle être une tâche plus complexe. Il est essentiel de trouver une méthode permettant au programme d'optimiser ces paramètres de manière autonome, car les nombreux filtres nécessaires pour cette tâche sont très dépendants des caractéristiques des images.

Les réseaux de neurones apparaissent actuellement comme la méthode la plus prometteuse pour détecter les mouvements. Leur capacité à s'adapter à divers types d'images et à apprendre les meilleures représentations pour la détection des doigts offre un potentiel significatif pour améliorer la précision et la robustesse des systèmes de détection.

Pour résumer, malgré les difficultés rencontrées, cette expérience a été enrichissante et nous a donné une meilleure compréhension des enjeux et des opportunités dans le domaine de la détection des mouvements. Dans le futur, nous aurions aimé approfondir l'étude des capacités des réseaux de neurones ainsi que dans la recherche de techniques d'optimisation automatique des paramètres afin d'assurer des performances fiables et évolutives.

Bibliographie

Références

- [1] *AdaBoost Algorithm in Machine Learning*. en. URL : <https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm> (visité le 03/05/2024).
- [2] Mahmoud AFIFI. “11K Hands : gender recognition and biometric identification using a large dataset of hand images”. In : *Multimedia Tools and Applications* (2019). DOI : 10.1007/s11042-019-7424-8. URL : <https://doi.org/10.1007/s11042-019-7424-8>.
- [3] *Animal Image Dataset (90 Different Animals)*. en. URL : <https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals> (visité le 03/05/2024).
- [4] *Cascade Classification — OpenCV 2.4.13.7 documentation*. URL : https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html (visité le 06/05/2024).
- [5] *Hand landmarks detection guide | MediaPipe*. en. URL : https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (visité le 03/05/2024).
- [6] HANDAGA. *handaga/tutorial-haartraining*. original-date : 2015-05-26T06:29:16Z. Déc. 2023. URL : <https://github.com/handaga/tutorial-haartraining> (visité le 03/05/2024).
- [7] Kalpana JOSHI et al. “Static Hand Gesture and Face Recognition System”. en. In : 9.9 (2021).
- [8] Aditya MITTAL. *Haar Cascades, Explained*. en. Jan. 2024. URL : <https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d> (visité le 03/05/2024).
- [9] *OpenCV : Cascade Classifier*. URL : https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html (visité le 03/05/2024).
- [10] P. VIOLA et M. JONES. “Rapid object detection using a boosted cascade of simple features”. en. In : *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. T. 1. Kauai, HI, USA : IEEE Comput. Soc, 2001, p. I-511–I-518. ISBN : 978-0-7695-1272-3. DOI : 10.1109/CVPR.2001.990517. URL : <http://ieeexplore.ieee.org/document/990517/> (visité le 03/05/2024).

Annexes



Déclaration sur l'honneur contre le plagiat

(à joindre obligatoirement à tout travail de recherche ou dossier remis à un enseignant)

Je soussigné(e),

Nom, Prénom,
.Kurth.Claire.....

Régulièrement inscrit à l'Université de Lorraine

N° de carte d'étudiant : 31500558.....

Année universitaire : 2023-2024.....

Niveau d'études : L ou M

Parcours : Informatique.....

N° UE : UE.811.....

Certifie qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Je certifie, de surcroît, que je n'ai ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Conformément à la loi, le non-respect de ces dispositions me rend passible de poursuites devant la commission disciplinaire et les tribunaux de la République Française.

Fait à , le ..6 mai 2024 à Nancy....

Signature :

A handwritten signature in black ink, appearing to read 'Claire Kurth'.

Annexes



Déclaration sur l'honneur contre le plagiat

(à joindre obligatoirement à tout travail de recherche ou dossier remis à un enseignant)

Je soussigné(e),

Nom, Prénom,
Dallé Victor.....

Régulièrement inscrit à l'Université de Lorraine

N° de carte d'étudiant : 31905179.....

Année universitaire : 2023-2024.....

Niveau d'études : L ou M

Parcours : Informatique.....

N° UE : UE 811.....

Certifie qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Je certifie, de surcroît, que je n'ai ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Conformément à la loi, le non-respect de ces dispositions me rend passible de poursuites devant la commission disciplinaire et les tribunaux de la République Française.

Fait à , le ..6 mai 2024 à Nancy....

Signature :

A handwritten signature in black ink, appearing to read 'Victor Dallé'.

Résumé

Ce projet a pour objectif de comprendre les mécanismes et les problématiques liés à la reconnaissance des gestes de la main. L'objectif principal étant de concevoir un système capable de détecter et de classifier différentes positions précises de la main effectuées par l'utilisateur telles que le poing fermé ou un certains nombres de doigts levés. Pour cela, nous avons exploré diverses méthodes de détection des mains et des doigts, en utilisant des classifieurs Haar-cascade et des techniques de traitement d'image (tels que l'utilisation de masques et de convex hull). Nous avons testé ces méthodes sur des images de mains et avons analysé les différents résultats obtenus. Nous avons également parlé des limites de ces méthodes employées et exploré des pistes d'amélioration pour optimiser la détection des gestes de la main. Enfin, nous avons abordé les enjeux et les opportunités des réseaux de neurones pour la détection des mouvements.

Summary

The aim of this project is to understand the mechanisms and issues involved in hand gesture recognition. The main objective is to design a system capable of detecting and classifying various precise hand positions performed by the user, such as a closed fist or a certain number of raised fingers. To achieve this, we explored various hand and finger detection methods, using Haar-cascade classifiers and image processing techniques (such as the use of masks and convex hull). We tested these methods on hand images and analyzed the different results obtained. We also discussed the limitations of the methods employed and explored avenues of improvement to optimize hand gesture detection. Finally, we briefly discuss the challenges and opportunities of neural networks for motion detection.