

Détection de signes de la main à partir d'une image de profondeur

Welcome Alexandre-Barff, Quentin Dorléat, Pierre Touboul, Alexis Vallet

8 janvier 2013

Table des matières

1	Présentation globale du programme	2
2	Segmentation	3
2.1	Présentation de la méthode	3
2.2	Analyse des résultats	7
3	Invariance par rotation et symétrie	11
3.1	Redressement de la main	11
3.1.1	Présentation de la méthode	11
3.1.2	Analyse	11
3.2	Détection de la direction du pouce	12
3.2.1	Présentation de la méthode	12
3.2.2	Analyse	12
4	Classification	12
4.1	Classifieurs par profils et histogrammes horizontaux et verticaux	12
4.1.1	Présentation de la méthode	12
4.1.2	Résultats	14
4.1.3	Améliorations possibles	16
4.2	Classifieur densités	16
4.2.1	Résultats	17
4.3	Classifieur histogramme radial	17
4.3.1	État de l'art	17
4.3.2	Présentation de la méthode	17
4.3.3	Analyse des résultats	20
4.4	Classifieur de convexité	22
4.4.1	Etat de l'art	22
4.4.2	Présentation	23
4.4.3	Tests : paramètres optimaux	26
4.4.4	Conclusion	29
4.4.5	Remarques	29

5	Combinaison des classifieurs	30
5.1	Description de la méthode	30
5.2	Analyse des résultats	30
6	Suivi de la main dans une séquence d'images	31
6.1	Présentation de la méthode	31
6.1.1	Segmentation de la main et extraction de la box	31
6.1.2	Opérateur de Harris	32
6.1.3	Seuillage sur la boîte englobante	32
6.1.4	Harris + Appariement des points de deux images seuillées	34
6.1.5	Déplacement de la box	35
6.1.6	Redimensionnement de la boîte	36
6.2	Analyse des résultats	36
6.3	Améliorations possibles	37

Introduction

Dans le cadre des UV IN52 et IN54, nous avons développé un programme nous permettant de mettre à profit les compétences acquises à travers ces UV. Le présent rapport fait état du travail réalisé. L'objectif du programme développé est la reconnaissance de forme au sein d'une image. Plus précisément, il s'agissait d'extraire une main au sein d'images en niveaux de gris de type carte de profondeur, puis de traiter ces images afin de déterminer le nombre de doigts que ces mains présentent.

Pour ce faire, nous avons effectué plusieurs traitements sur les images, que nous présenterons par parties. La première s'intéresse aux différents traitements effectués afin de segmenter l'image originale pour permettre son interprétation.

Dans une seconde partie, nous présenterons les classifieurs développés afin d'interpréter l'image segmentée, mais également les méthodes de décision associées aux différents classifieurs ainsi que les résultats obtenus par ces derniers.

Enfin, nous avons mis en œuvre une méthode permettant un suivi temporel d'une main dans une succession d'images. Les algorithmes développés seront présentés dans une dernière partie.

1 Présentation globale du programme

Le programme est divisé en 3 grandes étapes :

- L'étape de segmentation, consistant à extraire un rectangle englobant de la main à reconnaître, à la binariser et filtrer.
- L'étape de classification, consistant à déterminer le nombre de doigts de la main extraite par un certain nombre de classifieur.
- La combinaison, regroupant les résultats des classifieurs pour donner une estimation la plus fiable possible du nombre de doigts de la main.

De plus, un module de suivi de la main dans une séquence d'images a été développée pour effectuer un suivi de mouvement de la main.

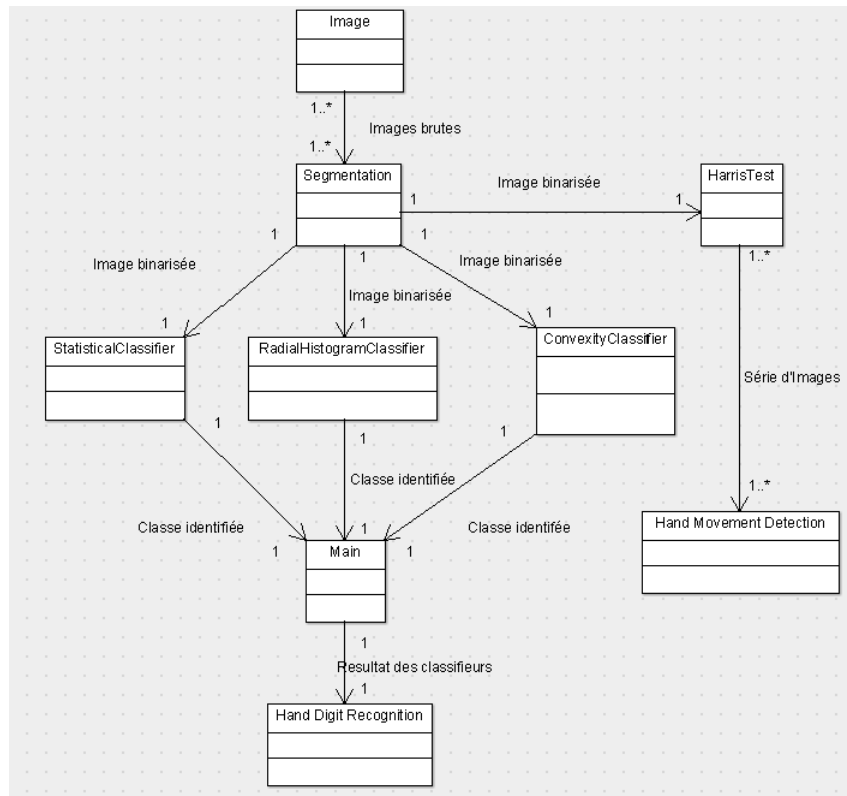


FIGURE 1 – Diagramme de classe simplifié du programme

2 Segmentation

2.1 Présentation de la méthode

Tout d’abord, il nous a fallu considérer les images initialement fournies. Les images fournies sont des cartes de profondeur en niveaux de gris. Ces images se trouvaient être au format YML, représentées sous forme matricielle par des nombres flottants. N’ayant trouvé de méthode OpenCV permettant le chargement de ces fichiers, et afin de faciliter le travail sur ces images, nous avons choisi de convertir les fichiers .YML originaux en fichier .BMP. Pour ce faire, nous avons développé une fonction permettant de charger les fichiers originaux, puis d’effectuer une conversion sur les données contenues afin de les rendre compatibles avec leur utilisation dans le cadre d’une image. Ainsi, nous avons tenté plusieurs conversions différentes afin de transformer les nombres flottants originaux pour obtenir des valeurs comprises entre 0 et 255.

Nous avons tout d’abord tenté une interpolation linéaire des valeurs originales. La formule de conversion était alors :

$$255 \times \frac{val - min}{max - min}$$

avec *val* le nombre flottant original, *min* la valeur minimale du fichier YML, *max* la valeur maximale du fichier YML.



FIGURE 2 – Résultat de l'interpolation linéaire des valeurs flottantes de l'image source.

Considérant ce résultat peu satisfaisant (Figure 2), nous avons tenté une seconde interpolation, basée non plus sur le minimum et maximum, mais sur la moyenne des valeurs du fichier YML original. Ainsi, la formule devint :

$$255 \times \frac{val}{moy}$$

A nouveau peu satisfait du résultat (Figure 3), nous avons tenté une troisième formule, avec un résultat élevé au carré afin de privilégier les zones foncées dont la main fait partie, au détriment des zones claires peu intéressantes dans notre cas (Figure 4). La conversion s'effectuait alors par :

$$(val \times 16)^2$$

La première étape de travail consiste à traiter les images initiales afin d'en extraire la zone d'intérêt à savoir ici la main. Travaillant sur des images de profondeur à une seule composante, la segmentation ne représente pas ici une étape compliquée du processus de traitement de l'image, mais ne peut non plus se borner à un simple seuillage binaire. En effet, dans la mesure où la main n'est pas toujours idéalement positionnée, mais sa distance à la caméra variant d'une image à l'autre, il est nécessaire de seuiller automatiquement l'image. En ce sens



FIGURE 3 – Résultat de l'interpolation basée sur la moyenne des valeurs.



FIGURE 4 – Résultat de la conversion quadratique.

et après plusieurs tests et quelques recherches, il s'est avéré que l'algorithme de Fisher répondait très bien à la problématique posée.

Etant disponible et efficace dans la librairie Pandore, nous avons entrepris d'utiliser cette dernière, ou tout du moins l'opérateur Fisher afin de ne pas redévelopper un algorithme implémenté de manière optimale. L'opérateur per-

met entre autre d'effectuer un multi-seuillage sur l'image et de répartir ses pixels selon les seuils déterminés.



FIGURE 5 – Image source et seuillée avec Fisher.

Comme on peut le constater, l'algorithme distribue les pixels selon n classes distinctes, n étant un paramètre de l'algorithme. Ce dernier a été fixé à 5 pour apporter des résultats optimaux sur le plus grand nombre d'images.

Une fois le multi-seuillage réalisé, nous bénéficions d'une image sur laquelle les pixels occupent n des 256 niveaux de gris. Partant du postulat que dans la plupart des cas présentés, l'élément à extraire se trouve en avant-plan de l'image de profondeur, cela signifie que la classe de pixels représentant cet élément est la plus sombre. Bien sûr, cela n'est pas une généralité et cela se vérifie uniquement lorsque la main est bien l'élément le plus sombre de l'image de profondeur, mais nous souhaitons tout d'abord avoir une bonne segmentation, efficace dans la majorité des cas présentés.

La seconde partie de la segmentation consiste à égaliser l'histogramme. De la sorte, la première classe de pixels prend la valeur 0 et tous les pixels de la main deviennent donc noirs. On peut alors effectuer une seuillage binaire sur la valeur 0 pour ne garder que la main (Figure 6).



FIGURE 6 – Entrée et sortie du seuillage binaire après égalisation de l'histogramme.

Comme on peut le constater sur les images ci-dessus, cette méthode n'est pas parfaite. Même si elle apporte des résultats confortables sur la plupart des images, elle rend en sortie une image sur laquelle on peut observer une quantité non négligeable de pixels résiduels. Nous avons donc développé une fonction de traitement permettant de « nettoyer » l'image et de supprimer l'ensemble de ces pixels épars qui ne font pas partie d'une composante connexe. Le principe est simple : Nous étudions la connexité de chaque pixel, afin de déterminer s'il appartient à un regroupement de pixels et donc s'il doit être supprimé.

Considérant $d_{inf}(P, Q)$, la distance de l'échiquier entre les points P et Q , on appelle voisinage d'ordre k du pixel P , l'ensemble des pixels Q définit par :

$$V_k(P) = \{Q \text{ tels que } 0 < d_{inf}(P, Q) < k\}$$

Considérant l'ordre de connexité 8, V_1 représente donc l'ensemble des 8 voisins directs, soit le plus petit voisinage non vide d'un pixel. L'opérateur de traitement consiste à supprimer un pixel P lorsque le nombre de pixels de la même couleur que P dans V_1 , est inférieur à une certaine limite notée α . C'est-à-dire que pour le pixel considéré, il faut au moins α pixels noirs dans la zone colorée en vert dans l'image ci-dessous (Figure 7).

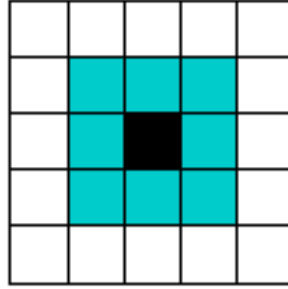


FIGURE 7 – Pixel connexité 8.

Après une batterie de tests, cette limite a été fixée à 5. En dessous, la main peut être entièrement supprimée après application du traitement, tandis qu'au dessus, certains pixels gênants pour la reconnaissance peuvent être laissés sur l'image. L'opérateur est appliqué plusieurs fois, ce qui permet d'opérer une érosion du poignet, gênant pour la reconnaissance. L'histogramme de l'image est inversé avant application de l'opérateur. Voici ci-dessous, le résultat de la segmentation sur la première image du groupe 1 (Figure 8).

2.2 Analyse des résultats

Cette méthode de segmentation se trouve être efficace sur la majorité des images fournies dans la base. Voici un panel de test réalisé sur l'ensemble des images du groupe 1 :

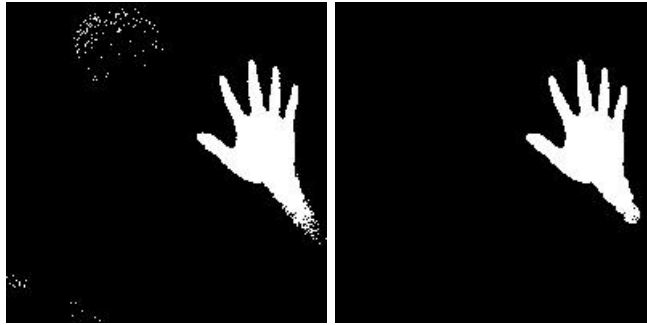
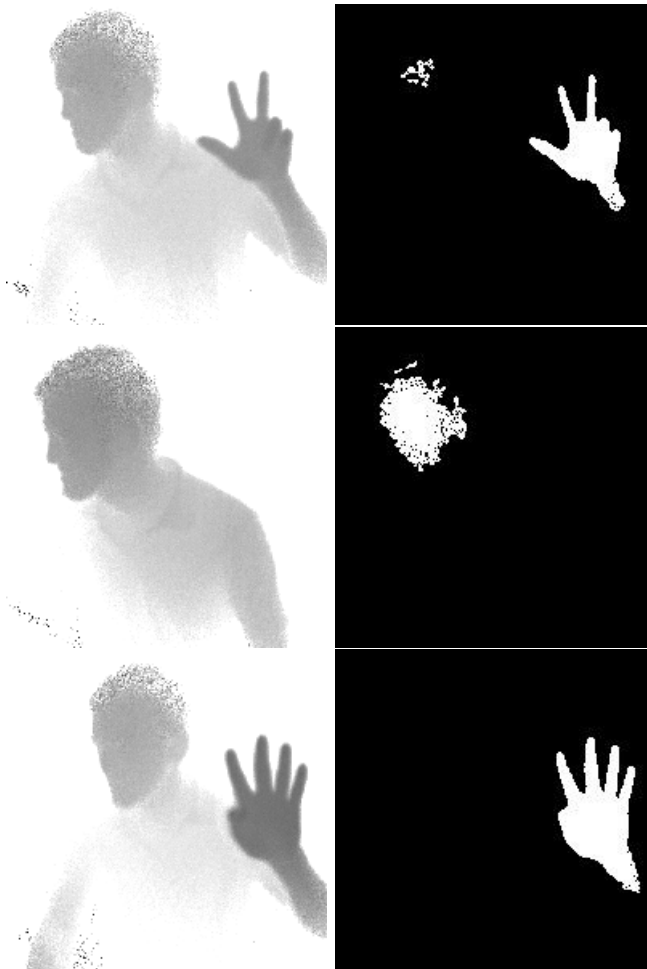


FIGURE 8 – Résultat de la segmentation pour la première image du groupe 1, avant et après filtrage.





Le taux de réussite s'avère très correct puisque 13 images sur 16 sont traitées correctement. On observe des erreurs dans deux cas :

- Lorsqu'il n'y a pas de main (2^{eme} image)
- Lorsque la main n'est pas en avant-plan (images 1 et 4).

Dans les deux cas, l'algorithme doit être en mesure de détecter que l'étape de segmentation n'a pas été optimale, dans la mesure où la reconnaissance ne renverra pas de résultat. L'algorithme de segmentation ne peut cependant pas être corrigé simplement pour palier les problèmes et lorsqu'une main est effectivement présente dans l'image, il n'est actuellement pas en mesure de se focaliser sur cette dernière pour l'extraire. Une telle solution n'a pas été implémentée par manque de temps et également cela demeure un cas très particulier. En effet dans une application de suivi de la main et de reconnaissance de geste, il est évident que l'homme qui effectue les gestes va instinctivement placer les mains devant son corps et non l'inverse.

Sur une autre base d'images qu'est le groupe 3, certaines images posent problème lorsque le bras est placé perpendiculairement à l'axe de vision de la caméra, autrement dit lorsque le bras et la main sont à égale distance de la caméra (images de 13 à 18 dans le groupe). Le contraste de l'image s'avère alors très peu élevé et des pixels résiduels noirs, bien plus foncés que le membre à reconnaître faussent alors l'étape de segmentation.

Dans le tableau ci-dessous, on observe un test sur 3 images. Comme on peut l'observer, la segmentation sur les deux premières est loin de fournir un résultat satisfait exploitable par la reconnaissance, tandis que la dernière est segmentée « parfaitement » sans correction nécessaire. Une solution a consisté à nettoyer l'image en prétraitement, afin de supprimer les pixels résiduels sombres, faussant la segmentation sur 5 des 18 images du groupe 3. Seulement, comme on peut le constater, cette correction s'avère être néfaste sur des images sur laquelle la segmentation était initialement réussie. Nous n'avons à l'heure actuelle pas exploré de piste nous permettant de traiter idéalement et automatiquement toutes les images.


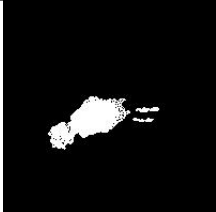


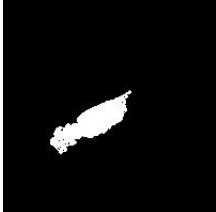


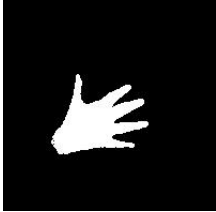

Image source	Image non corrigée avant segmentation				Image corrigée avant segmentation			
								
								
								

FIGURE 9 – Résultats de la correction.

3 Invariance par rotation et symétrie

3.1 Redressement de la main

3.1.1 Présentation de la méthode

Un des premiers problèmes que nous avons rencontré est lié à la rotation des mains, certaines mains ne sont pas orientées verticalement alors que nos classifieurs font l'hypothèse que toutes les mains sont orientées verticalement (c'est à dire avec les doigts vers le haut). Pour redresser la main, il nous fallait tout d'abord déterminer la direction de la main dans l'image. Pour ceci, nous avons utilisé une technique inspirée de l'analyse en composantes principales pour estimer la direction "principale" de l'image.

Soit I une image segmentée et binarisée d'une main, définissons P l'ensemble des positions des pixels appartenant à la main dans I . Nous déterminons alors la matrice de covariance de ces points par rapport à leur centre de gravité (qui correspond à la valeur moyenne des points), et nous considérons les 2 vecteurs propres de cette matrice. Nous considérons que l'un d'entre eux est la direction de la main (Figure 10). Nous appliquons alors une rotation à l'image segmentée binarisée de façon à obtenir la verticale dans la direction de ce vecteur propre. Le calcul a été implémentée à l'aide de la classe PCA d'OpenCV.

3.1.2 Analyse

Cette méthode a 2 défauts :

- La direction donnée pointe parfois vers le poignet au lieu des doigts. Ce problème peut être résolu par une méthode invariante par rotation d'angle π , comme l'histogramme radial.
- La méthode donne de mauvais résultats lorsqu'aucun doigt n'est levé.

Cependant, les images mal redressées par cette méthode sont peu nombreuses (moins de 10%), cette méthode rend donc possible l'utilisation de méthodes de classification comme l'histogramme radial.



FIGURE 10 – Image segmentée et binarisée d'une main, avec en vert et rouge les deux vecteurs propres de la matrice de covariance.

3.2 Détection de la direction du pouce

3.2.1 Présentation de la méthode

Un autre problème rencontré est du au mélange de mains gauche et droite dans les données, ainsi que de mains de dos et mains de face. En effet, avec les même doigts levés, ces images n'ont pas les même caractéristiques. Pour ceci, nous estimons la direction des doigts de la main en calculant le décalage horizontal et vertical du centre de gravité de la main avec le centre de la paume de la main. Nous calculons le centre de la paume en supprimant les doigts de l'image binarisée par érosion avec un élément structurant suffisamment grand pour supprimer entièrement les doigts de la main (Figure 11). Si le décalage est supérieur à un $\epsilon > 0$ fixe, alors nous sommes capable de déterminer le "sens" de la main.

3.2.2 Analyse

Cette méthode ne permet pas cependant de conclure dans tous les cas, notamment lorsque le décalage entre les centres est trop faible (dans le cas d'une main sans doigts levés, ou sans pouce levé par exemple). Il est alors préférable d'utiliser une méthode de classification invariante par symétrie d'axe vertical comme l'histogramme de projection horizontal.



FIGURE 11 – Image segmentée et binarisée d'une main, avec en vert le centre de la paume de la main et en rouge le centre de gravité de la main.

4 Classification

4.1 Classifieurs par profils et histogrammes horizontaux et verticaux

4.1.1 Présentation de la méthode

Le premier classifieur développé se base sur la détermination de vecteurs caractéristiques de la main à reconnaître. Il a été subdivisé en deux sous classes, dont chacune se base sur une caractéristique différente. Ce classifieur travaille sur les images des mains segmentées et extraites de l'image originale.

La première classe s'intéresse aux profils de la main à détecter, soit la distance entre les bordures gauches et droites, hautes et basses, et le premier

pixel constituant la main à traiter, et ce pour un certain nombre de lignes régulièrement espacées. Sur le même principe, les profils verticaux sont générés.



FIGURE 12 – Profils verticaux d'une main.



FIGURE 13 – Profils horizontaux d'une main.

Ainsi, on détermine la morphologie globale de la main à traiter. Il est à noter que tous les profils sont normalisés (par division par la largeur en pixel de chaque image traitée), et ce afin de pouvoir être comparés entre eux, chaque main traitée ayant des dimensions différentes.

La seconde classe quant à elle s'intéresse aux histogrammes. Pour un certain nombre de lignes, on compte le nombre de pixels constitutifs de la main à traiter (dans notre cas, les pixels blancs), afin de déterminer non pas la morphologie de la main à reconnaître, mais la densité de la ligne considérée. Selon le même principe, l'histogramme d'un certain nombre de colonnes est calculé. Ces histogrammes sont également normalisés afin de pouvoir comparer des images de dimensions différentes.

Le nombre de lignes à considérer est paramétrable à travers la variable `NB.PROFILES`. Cette variable représente en effet le nombre de lignes et de colonnes à considérer pour déterminer les profils ou les histogrammes horizontaux et verticaux. Plus le nombre de lignes considérées est important, plus le profil global du caractère sera précis. Par conséquent, ce paramètre a une influence directe très importante sur le taux de reconnaissance des mains, mais également sur le coût d'exécution. Si une valeur trop basse est choisie, la détection de la morphologie de la main risque d'être trop peu significative pour obtenir des résultats probants. À l'inverse si une valeur trop élevée est choisie, la différence entre deux mains d'une même classe sera évaluée comme étant importante. Par ailleurs, l'augmentation du nombre de lignes à évaluer induit également une augmentation du temps de traitement des images.

Afin de disposer de points de comparaisons pour l'image à traiter, on établit une base de référence, subdivisée en un certain nombre de classes représentant chacune un nombre de doigts. Basiquement, la base est constituée de 6 classes, numérotées de 0 à 5 et constituées chacune d'un certain nombre d'images de références présentant le nombre de doigts correspondant à la classe en question. Pour établir cette base, on traite toutes les images de chaque classe afin de déterminer les vecteurs caractéristiques, profils ou histogrammes. Une fois toutes les images de la base traitées, on établit une moyenne des profils ou des histogrammes de chaque classe, afin d'obtenir le vecteur caractéristique de chaque classe de la base. C'est à partir de ces vecteurs moyens que chaque image à reconnaître sera comparée.

La comparaison s'effectue par distance euclidienne entre le vecteur caractéristique moyen de chaque classe et le vecteur caractéristique de l'image à reconnaître. Les vecteurs caractéristiques horizontaux et verticaux sont combinés en un seul par simple concaténation. On obtient ainsi la distance de l'image traitée à chacune des classes. Plus cette distance est faible, plus l'image à reconnaître présente de similarités avec la classe en question. Ces distances à chaque classe sont par ailleurs transformées en taux de probabilité, à travers la formule :

$$P(\omega_i/x) = \frac{\exp(-d(x, \omega_i))}{\sum_{j=0}^k -d(x, \omega_j)}$$

où ω_i est la classe i , x est l'image traitée et d est la distance euclidienne.

Il est à noter que les 6 classes de la base, représentant le nombre de doigts, peuvent être subdivisées en sous classes. En effet, pour un même nombre de doigts, une main peut présenter des configurations différentes. Par exemple, pour présenter deux doigts, il est fréquent de rencontrer deux configurations de la main : la première présentant le pouce et l'index, la seconde présentant l'index et le majeur. Hors, ces deux configurations différentes pour un même nombre de doigt présentent des vecteurs caractéristiques très différent l'un de l'autre. C'est la raison pour laquelle il est possible de distinguer ces configurations de la main au sein de différentes classes, auxquelles on associe le nombre de doigts correspondant.

Par ailleurs, une seconde méthode de prise de décision a été implémentée, celle des K plus proches voisins. Pour ce faire, le vecteur caractéristique de l'image à traiter est comparé non plus avec le vecteur caractéristique moyen de chaque classe, mais avec ceux de l'intégralité des images constitutives de la base d'apprentissage. On retient ainsi les K images les plus proches, puis nous calculons les probabilités d'appartenance aux classes de la main à traiter par représentation de chaque classe parmi les K plus proches.

4.1.2 Résultats

Les résultats obtenus sont très aléatoires, en fonction des différents paramètres intervenant. Ainsi, d'une base d'apprentissage à une autre, d'une image testée à une autre, le taux de réussite oscille entre 20% et plus de 80%, en atteignant parfois 100% pour certaines séries d'images. Cependant les résultats

restent très irréguliers, et restent en moyennent autours de 50% de réussite. Par ailleurs, une combinaison de ces deux classifieurs donne également des résultats très irréguliers, parfois améliorant quelque peu le taux de détection, parfois le dégradant. Nous pensons que la base d'apprentissage constituée reste trop restreinte pour permettre d'obtenir des résultats probants.

Dans tous les résultats suivant, voici la correspondance des différents paramètres :

- Base d'apprentissage 1 : une base d'apprentissage simple, composée de 6 classes, et constituée d'images présentant une configuration assez simple.
- Base d'apprentissage 2 : une base d'apprentissage étendue, composée de 12 classes en différenciant certaines configuration de la main pour un même nombre de doigts.
- Série 1 : les images de la base d'apprentissage 1.
- Série 2 : les images de la base d'apprentissage 2.
- Le paramètre "redressées" indique si les images ont subi un redressement ou non.

Voici une série de résultats des classifieurs par profils et par histogramme, en fonction de la base d'apprentissage, de la série d'images testée, et du nombre de lignes considérées : (Figure 14).

Base apprentissage	Série de test	redressées		PROFILS	HISTO
Base 1	Série 2	oui	D=12	51.00%	48.00%
Base 1	Série 2	non	D=12	51.00%	53.00%
Base 2	Série 1	oui	D=12	66.00%	66.00%
Base 2	Série 1	non	D=12	100.00%	66.00%
Base 1	Série 2	oui	D=10	48.00%	55.00%
Base 1	Série 2	non	D=10	44.00%	46.00%
Base 2	Série 1	oui	D=10	83.00%	66.00%
Base 2	Série 1	non	D=10	83.00%	66.00%
Base 1	Série 2	oui	D=8	51.00%	60.00%
Base 1	Série 2	non	D=8	51.00%	55.00%
Base 2	Série 1	oui	D=8	83.00%	50.00%
Base 2	Série 1	non	D=8	66.00%	66.00%
Base 1	Série 2	oui	D=6	58.00%	53.00%
Base 1	Série 2	non	D=6	46.00%	51.00%
Base 2	Série 1	oui	D=6	66.00%	50.00%
Base 2	Série 1	non	D=6	66.00%	50.00%
Base 1	Série 2	oui	D=4	41.00%	41.00%
Base 1	Série 2	non	D=4	46.00%	46.00%
Base 2	Série 1	oui	D=4	66.00%	50.00%
Base 2	Série 1	non	D=4	83.00%	50.00%

FIGURE 14 – Résultats du classifieur profils et histogrammes verticaux/horizontaux avec distance Euclidienne.

Considérant que le meilleur compromis est atteint pour $D = 8$, voici une seconde série de résultats obtenus par la méthode des KPPV, K variant de 3 à 6 : (Figure 15).

			Feuille1			
HISTO, KPPV			K = 3	K = 4	K = 5	K = 6
Base 1	Série 2	oui	49.00%	49.00%	49.00%	44.00%
Base 1	Série 2	non	39.00%	42.00%	39.00%	47.00%
Base 2	Série 1	oui	50.00%	66.00%	66.00%	50.00%
Base 2	Série 1	non	50.00%	66.00%	66.00%	50.00%
PROFILS KPPV			K = 3	K = 4	K = 5	K = 6
Base 1	Série 2	oui	37.00%	32.00%	44.00%	46.00%
Base 1	Série 2	non	35.00%	39.00%	32.00%	32.00%
Base 2	Série 1	oui	33.00%	33.00%	33.00%	33.00%
Base 2	Série 1	non	66.00%	66.00%	50.00%	50.00%

FIGURE 15 – Résultats du classifieur profils et histogrammes verticaux/horizontaux avec KPPV.

4.1.3 Améliorations possibles

L'amélioration la plus importante à apporter est probablement l'enrichissement de la base d'apprentissage. En effet, nos bases actuelles disposent de trop peu d'images, ce qui génère des différences importantes entre les images de la base et certaines configurations hasardeuses de la main à tester.

Une seconde amélioration envisagée serait de pondérer la prise en compte des caractéristiques des différentes zones de l'image à tester, notamment en augmentant le poids des mesures effectuées sur la partie supérieure de la main, là où se trouvent les doigts. A l'inverse, il serait intéressant de réduire l'importance des mesures effectuées dans la partie basse de l'image, zone dans laquelle se trouve le poignet que la segmentation ne permet pas toujours d'éliminer. Cependant, il faudrait pour ce faire disposer d'une fonction de redressement des mains moins hasardeuse. En effet, certaines images sont tout bonnement redressées à l'envers, ce qui fausse la comparaison par la suite.

Enfin, la méthode de décision basée sur les KPPV peut également être améliorée. Dans le cas où plusieurs classes seraient équi-représentées, la classe actuellement retournée est la première ayant atteint la représentation maximale. Il serait judicieux de retourner la classe représentée dont la distance à une image de la base est la plus faible.

4.2 Classifieur densités

Le second classifieur implémenté s'intéresse à la densité de l'image à traiter. Chaque image (après segmentation et extraction de la main) est subdivisée en plusieurs zones de tailles égales. Puis, pour chaque zone, on détermine le nombre de pixels appartenant à la main (dans notre cas les pixels blanc). Ce nombre de

pixel est ensuite normalisé, par division par le nombre total de pixel de la zone. On obtient ainsi la densité des différentes zones de l'image.

Sur le même principe que pour les classifieurs précédents, on établit une base d'apprentissage divisée en plusieurs classes correspondant au nombre de doigts. Chaque classe de la base est constituée d'un certain nombre d'images présentant des mains du nombre de doigts correspondant à la classe. Chaque image à traiter est comparée avec cette base, et l'on obtient un vecteur de probabilité de chaque classe sur le même modèle que précédemment.

Par ailleurs, le nombre de subdivisions de l'image est également paramétrable via les variables M et N. A nouveau, ce nombre de zones a une influence importante sur le taux de détection. Un nombre trop faible ne permettra pas d'obtenir une carte de densité suffisamment précise pour obtenir des résultats satisfaisants. A l'inverse, un trop grand nombre de zones peut produire des écarts importants, et le taux de reconnaissance se dégrade.

4.2.1 Résultats

Base apprentissage	Série de test	redressées	N=M=2	N=M=3	N=M=4	N=M=5	N=M=6	N=M=7	N=M=8	N=M=9	N=M=10
Base 1	Série 2	oui	34.00%	37.00%	37.00%	42.00%	40.00%	44.00%	42.00%	39.00%	42.00%
Base 1	Série 2	non	32.00%	35.00%	30.00%	37.00%	33.00%	26.00%	37.00%	37.00%	35.00%
Base 2	Série 1	oui	16.00%	33.00%	50.00%	33.00%	50.00%	50.00%	50.00%	50.00%	50.00%
Base 2	Série 1	non	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%

FIGURE 16 – Résultats du classifieur densités avec distance euclidienne.

Base apprentissage	Série de test	redressées	K=3	K=4	K=5	K=6
Base 1	Série 2	oui	44.00%	46.00%	39.00%	35.00%
Base 1	Série 2	non	35.00%	37.00%	32.00%	30.00%
Base 2	Série 1	oui	33.00%	33.00%	16.00%	16.00%
Base 2	Série 1	non	50.00%	50.00%	33.00%	33.00%

FIGURE 17 – Résultats du classifieur densités avec KPPV.

4.3 Classifieur histogramme radial

4.3.1 État de l'art

Cette technique avait été utilisée avec succès pour la reconnaissance de signes de la main avec une caméra Kinect[1], c'est pourquoi nous avons jugé intéressant de l'implémenter.

4.3.2 Présentation de la méthode

Cette méthode de classification consiste à extraire comme vecteur caractéristique de la main un histogramme radial pondéré des pixels de la main autour d'un

centre, pour ensuite classifier ce vecteur à l'aide de classifieurs Bayésiens, K plus proches voisins ou réseau de neurone à une couche cachée.

Calcul de l'histogramme radial Intuitivement, l'histogramme radial consiste à calculer un histogramme de projection des pixels de la main "autour" du centre de gravité de la main. Pour déterminer celui-ci, définissons tout d'abord I l'image de la main segmentée et binarisée :

$$I(x, y) = \begin{cases} 1 & \text{si } (x, y) \text{ est un pixel de la main} \\ 0 & \text{sinon} \end{cases}$$

Nous déterminons ensuite une image I' de décalage d'angles, qui associe, pour chaque droite passant par le centre de gravité $c = (x_c, y_c)$ de la main, un niveau de gris distinct. C'est en calculant l'histogramme des niveaux de gris de cette image que nous obtenons l'histogramme radial. Ainsi, nous définissons I' de la façon suivante pour tout (x, y) de la main :

$$I'(x, y) = \tan^{-1}\left(\frac{x - x_c}{y - y_c}\right)$$



FIGURE 18 – Image de décalages d'angles

Nous calculons alors l'histogramme des niveaux de gris de cette image, et nous obtenons alors des histogrammes donnant des pics distincts pour chaque doigt de la main (Figure 19). Il est important de noter que l'histogramme radial est invariant par rotation de centre le centre de gravité et d'angle π , ce qui nous permet de classifier la main même lorsqu'elle se retrouve à pointer vers le bas après redressement.

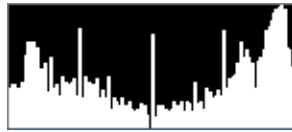


FIGURE 19 – Histogramme radial avec centre de gravité.

Cependant, le centre de gravité a tendance à être positionné là où le pouce et l'auriculaire ont le même niveau de gris dans l'image de décalages d'angles. C'est pourquoi nous utilisons à la place du centre de gravité de la main le centre de



FIGURE 20 – Histogramme radial avec centre de la paume)

la paume de la main. Celui-ci étant placé plus bas sur la main, celui-ci produit des pics plus distincts pour chaque doigt dans l'histogramme radial (Figure 20).

L'histogramme radial ainsi obtenu constitue le vecteur caractéristique de la main. Nous avons utilisé plusieurs méthodes pour classifier ce vecteur caractéristique :

- Classifieur Bayésien simple, sans spécificité particulière. Nous avons pour cela utilisé la classe `CvNormalBayesClassifier` du module machine learning d'OpenCV.
- K plus proches voisins, encore une fois sans spécificités. Nous avons pour cela utilisé la classe `CvKNearestClassifier` du module machine learning d'OpenCV.
- Réseau de neurones artificiels, avec une topologie à 3 couche :
 - Une couche d'entrée avec n neurones, correspondants aux n classes de l'histogramme radial.
 - Une couche cachée avec k neurones, où k est variable selon le compromis précision/efficacité souhaité.
 - Une couche de sortie avec m neurones, où m est le nombre de classes dans lesquelles nous souhaitons classifier l'histogramme (généralement 6, une pour chaque nombre de doigts possibles). La valeur maximale du vecteur de sortie correspond à la classe choisie.

Calcul de sous-classes automatique par K moyennes Un problème rencontré est que des images de mains comportant le même nombre de doigts peuvent avoir des histogrammes radiaux très différents. Cela est dû entre autre au nombre de façon différentes de lever 3 doigts, par exemple. Cela pose problème dans le cadre du classifieur Bayésien et du classifieur avec réseaux de neurones. Pour ceci, nous avons évalué plusieurs approches :

- Séparer chaque classe en sous-classe, de telle façon à ce que chaque sous-classe corresponde à une configuration de la main donnée. Ainsi, nous obtenons 32 classes distinctes (1 bit pour chaque doigt). Cette méthode s'est prouvée infructueuse étant donné le manque de données d'apprentissage pour la plupart des configurations de doigts.
- Utiliser un algorithme d'apprentissage non supervisé pour classer les histogrammes radiaux de mains de la base d'apprentissage ayant le même nombre de doigts dans des sous-classes d'histogrammes radiaux proches. Cette deuxième approche s'est montrée efficace pour augmenter le taux de reconnaissance avec le classifieur Bayésien.

4.3.3 Analyse des résultats

Mesure du taux de reconnaissance Étant donné la petite taille des données disponibles, il n'a pas été fructueux de séparer les images dans une base d'apprentissage (servant à entraîner les classifieurs Bayésiens, KPPV et réseau de neurone) et une base de test (servant à mesurer le taux de reconnaissance) uniques et distinctes. Nous avons résolu le problème en mesurant le taux de reconnaissance par la méthode du "leave one out" :

Soit $Q = \{(image, nbdoigts), \dots\}$ une base contenant toutes les images disponibles.

- Initialiser une variable c à 0.
- Pour chaque $q_i = (image_i, nb_i) \in Q$, faire :
 - Utiliser $Q - \{q_i\}$ comme base d'apprentissage pour le classifieur
 - Si le classifieur retourne nb_i pour $image_i$, alors incrémenter c
- Le taux de reconnaissance est alors $\frac{c}{|Q|}$

C'est par cette méthode que le taux de reconnaissance est mesuré dans les paragraphes suivants.

Classifieur Bayésien et choix du nombre de classes pour l'histogramme

Un paramètre important est le choix du nombre de classes pour l'histogramme radial de la main. Pour cela, nous avons mesuré le taux de reconnaissance en fonction du nombre de classes de l'histogramme avec le classifieur Bayésien, qui ne nécessite pas de paramètres supplémentaires.

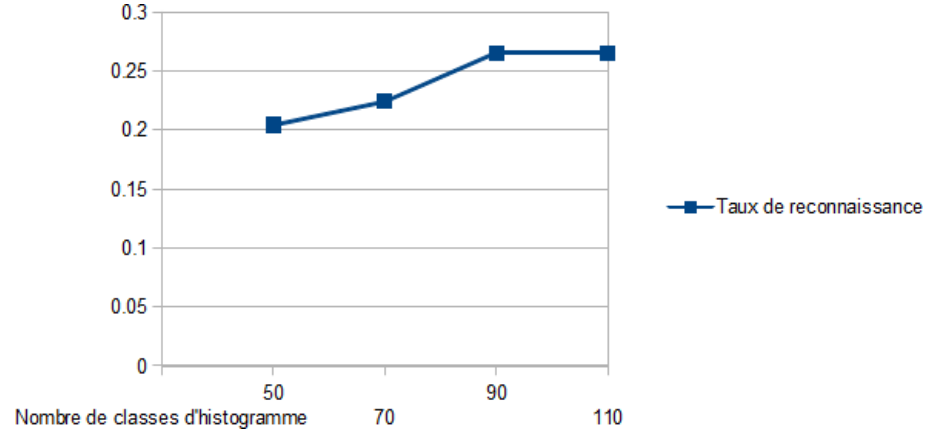


FIGURE 21 – Taux de reconnaissance par rapport au nombre de classes de l'histogramme avec classifieur de Bayes.

Le taux de reconnaissance croît avec le nombre de classes de l'histogramme, jusqu'à atteindre un plafond à 90 classes d'histogramme environ (Figure 21). Ajouter des sous-classes calculées avec l'algorithme des K moyennes permet d'augmenter sensiblement ce taux (Figure 22). Cependant cela rajoute un élément aléatoire, lié à l'initialisation des sous-classes.

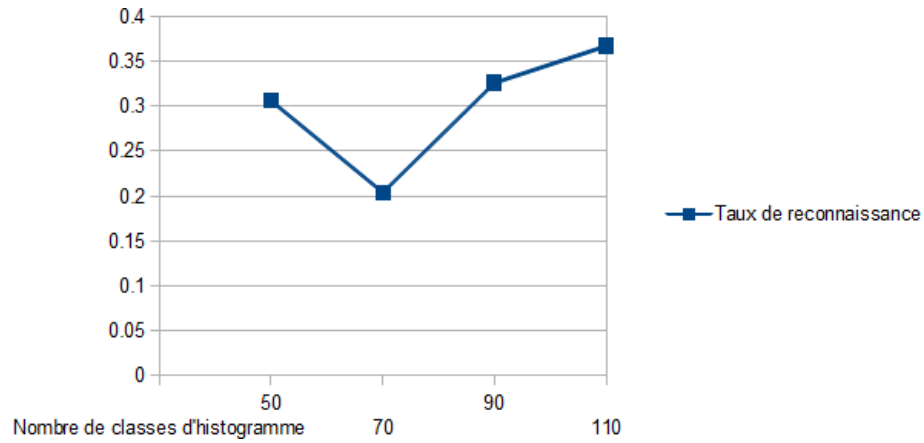


FIGURE 22 – Taux de reconnaissance par rapport au nombre de classes de l’histogramme avec 2 sous-classes par classe.

Classifieur K plus proches voisins En fixant le nombre de classes de l’histogramme à 90, et en faisant varier K , nous obtenons un taux de reconnaissance maximal de 38% avec $K = 7$ (Figure 23).

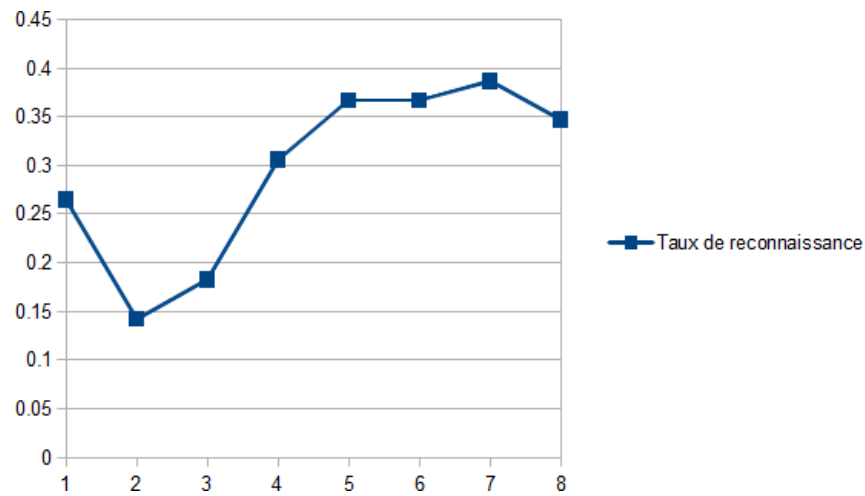


FIGURE 23 – Taux de reconnaissance par rapport à K avec les K plus proches voisins

Classifieur avec réseau de neurones artificiels En fixant de nouveau le nombre de classes de l’histogramme à 90, et en faisant varier le nombre de neurones de l’unique couche cachée, nous obtenons un taux de reconnaissance maximal de 26% avec 450 neurones dans la couche cachée. Il semble difficile

d'obtenir un taux de reconnaissance supérieur à un choix aléatoire uniforme (Figure 24).

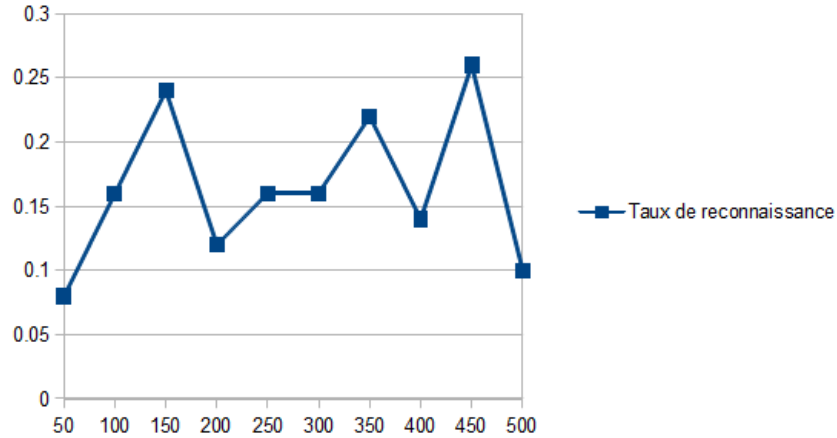


FIGURE 24 – Taux de reconnaissance par rapport au nombre de neurones dans la couche cachée

Conclusion, problèmes et améliorations possibles Après mesure des résultats, nous avons décidé d'utiliser le classifieur avec K plus proches voisins car celui-ci offre à la fois le meilleur taux de reconnaissance et les meilleures performances en terme de temps de calcul. Cependant, le taux de reconnaissance n'atteint que 38%, ce qui montre plusieurs problèmes avec cette approche :

- Malgré le redressement de l'image, certaines images sont mal redressées et sont donc mal classifiées.
- Comme les pics abrupts le montre sur (Figure 20), cette méthode est sensible au bruit dans l'image, et également à la présence ou non du poignet après la segmentation.
- Le nombre d'image disponibles pour effectuer l'apprentissage est limité, et toutes les configurations de doigts ne sont pas représentées ou seulement une fois. On peut raisonnablement espérer un meilleur taux de reconnaissance avec une plus grande base d'apprentissage.

4.4 Classifieur de convexité

4.4.1 Etat de l'art

Ce classifieur est basé sur le rapport « Hand Gesture Recognition Using Kinect » de Heng Du et TszHang To de l'Université de Boston [2]. Leurs travaux ont portés sur la reconnaissance de nombre de doigts à partir d'une image capturée à l'aide d'une caméra kinect et de la bibliothèque OpenCV.

4.4.2 Présentation

Le principe général de ce classifieur est de détecter les points de convexités et de concavités du contour de la main et ainsi, selon un raisonnement logique, calculer le nombre de doigts levés.

Dans le but de rajouter un classifieur ne nécessitant pas de base d'apprentissage et ainsi diversifier le nombre d'images, ainsi que la diversité d'approches pour détecter et reconnaître les signes statiques de la main, ce classifieur a été implémenté suivant les étapes suivantes :

Détection du contour de la main Une fois une sous-image de la main extraite de l'image d'origine puis segmentée, nous avons utilisé la fonction implémentée d'OpenCV `findContours()` afin de pouvoir ainsi dérouler l'algorithme uniquement sur le contour de la main, comme illustrée par la figure suivante : (Figure 25).

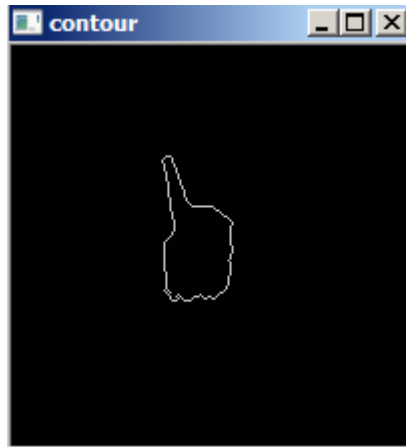


FIGURE 25 – Contour de la main par `findContours`.

Approximation de la forme de la main par un polygone L'étape suivante permet de simplifier la détection des points de convexité et de concavité. Pour cela, on utilise la fonction implémentée d'OpenCV `approxPolyDP()` basée sur l'algorithme récursif de Ramer-Douglas-Peucker qui réduit le nombre de points d'une courbe en approximant une courbe similaire contenant moins de points, selon les étapes suivantes :

1. Traçage d'un segment entre le premier point de la courbe et le dernier.
2. Recherche du point le plus éloigné à ce segment et appartenant à la courbe et marquage de ce point comme visité.
3. Comparaison de distance entre ce point et le segment :

- (a) Si la distance est inférieure à α (> 0 une valeur déterminée), alors on retire tous les points non marqués.
 - (b) Si la distance est supérieure à α , alors on marque le point et on trace le nouveau segment contenant le premier point et le dernier passant par le nouveau point marqué.
4. On applique l'algorithme récursivement sur les deux segments : (premier point, nouveau point) et (nouveau point, dernier point), jusqu'à ce qu'on soit limité par le quadrillage pixellaire (deux points ont la même coordonnée dans l'image).
5. Une fois la récursion terminée, on retrace la nouvelle courbe avec les points visités uniquement.

Détection des points convexes du polygone Afin de détecter les points de convexité de la main, on calcul un contour entourant le polygone en utilisant la fonction implémentée d'OpenCV `convexHull()`. Comme illustrée par la figure suivante : (Figure 26).



FIGURE 26 – Polygone englobant de la main tel que calculé par `convexHull`.

Détection des points de concavités du polygone En utilisant le résultat de la détection des points convexes, on effectue à présent la détection des points de concavité en utilisant une fonction adaptée d'OpenCV `cvConvexityDefects()`. Comme illustrée par la figure suivante : (Figure 27).

Filtrage des points de convexités et de concavités L'avant dernière étape de l'algorithme est l'étape la plus importante, car elle est assujettie à des

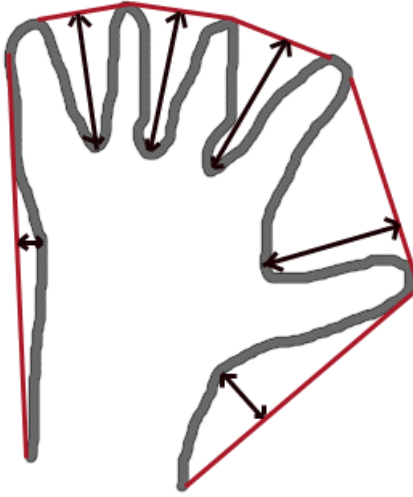


FIGURE 27 – Points de concavité tels que détectés par cvConvexityDefects.

paramètres statiques prédéterminés avant le lancement de cette dernière. Tout d’abord nous calculons un rectangle approximatif de la paume de la main en utilisant la fonction implémentée d’OpenCV `minAreaRect()`. Par la suite, afin de filtrer les points de convexité et respectivement de concavité, on compare leurs coordonnées en y avec le centre approximatif de la paume $+ \epsilon$ (> 0 valeur donnée) et respectivement β . Ces dernières nous permettent d’affiner nos résultats en augmentant la coordonnée en y du centre approximatif de la paume et ainsi réduire le nombre de points gardés.

Puis une fois les points de concavité et de convexité filtrés, on filtre une dernière fois les points de convexité selon leurs distances euclidiennes entre elles pour ainsi déterminer potentiellement le bout des doigts selon une distance minimale μ .

Calcul du résultat du classifieur Enfin la dernière étape est un raisonnement logique selon le nombre de points trouvés. En effet le résultat du classifieur est donné par le nombre de points convexes identifiés :

- Si le nombre de points concaves > 0 , Alors le résultat est le nombre de points convexes
- Sinon, On regarde s’il existe un point convexe qui respecte la distance minimum entre le point de convexité et le centre de la paume :
 - Alors le résultat est 1
 - Sinon le résultat est 0

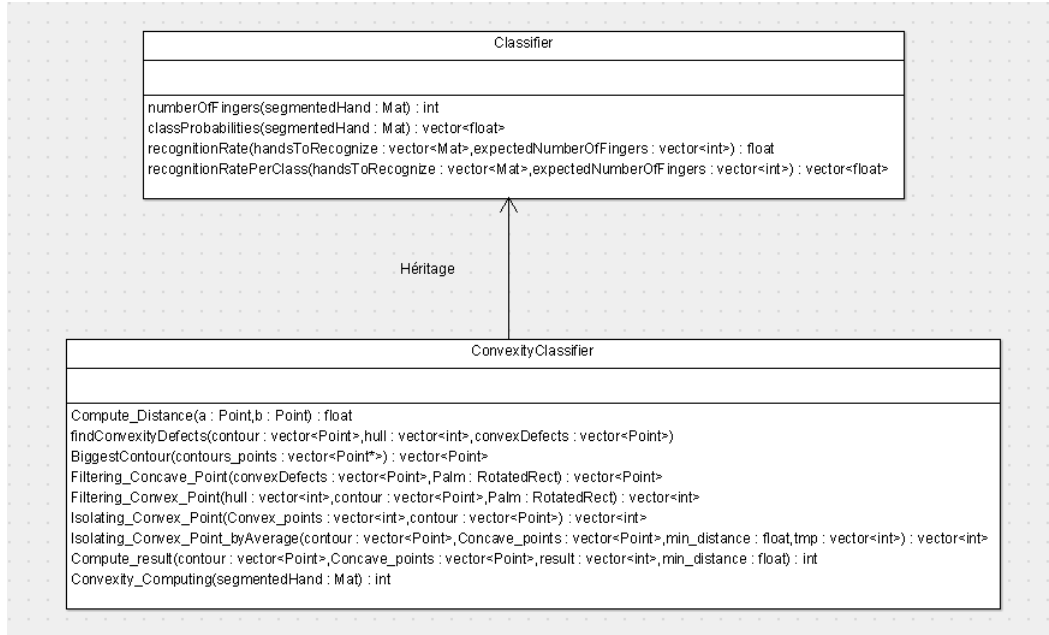


FIGURE 28 – Diagramme UML du classifieur par convexités.

4.4.3 Tests : paramètres optimaux

Parce que les zones des images traitées sont différentes, 3 paramètres nécessitent d'être statiques car on ne peut prévoir à l'avance la direction de la main ni même sa distance par rapport à la caméra. Dans ce principe, on a effectué une série d'essais pour déterminer des valeurs optimales pour les 3 paramètres :

- Distance minimale des points de convexités : ϵ
- Distance minimale des points de concavités : β
- Distance minimale entre points de convexités : μ

On remarque que le taux de reconnaissance atteint son maximum sur une plage de valeurs de [25, 25, 12] (Figure 29).

A partir de ces tests préliminaires, on a peu réduire les intervalles de valeurs afin de déterminer les paramètres individuellement.

Détermination de la distance minimale des points de convexité On remarque que le taux de reconnaissance s'améliore dans une fourchette de valeurs entre [27 , 20] et décroît en flèche en dehors de cet intervalle (Figure 30).

Cet intervalle représente la valeur moyenne d'écart entre le centre de la paume des mains dans la base d'apprentissage et les points de convexité, ce qui expliquerait ainsi cette amélioration du taux de reconnaissance. En dessous, les points de convexité à considérer seront trop importants et au dessus le nombre de points de convexité sera insuffisant pour pouvoir déterminer avec précision

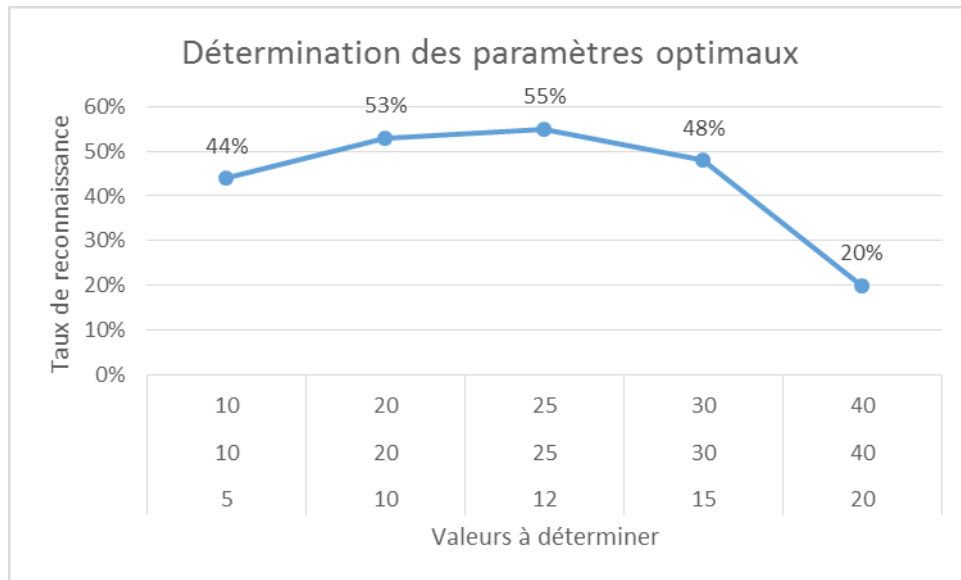


FIGURE 29 – Taux de reconnaissance par rapport aux paramètres ϵ , β et μ .

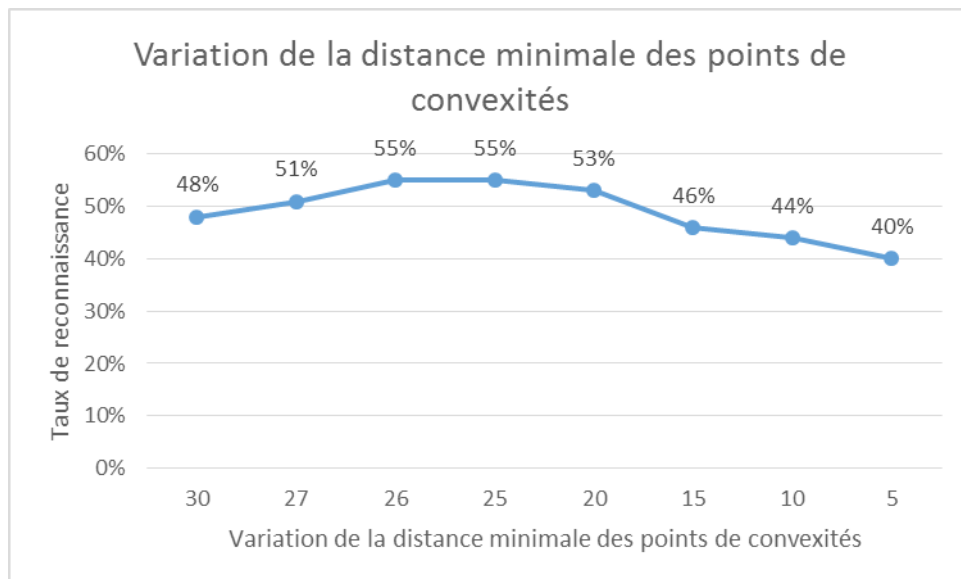


FIGURE 30 – Taux de reconnaissance par rapport à la variation de distance minimale des points de convexité.

le résultat attendu. On obtient alors une distance optimale de $\epsilon = 25$.

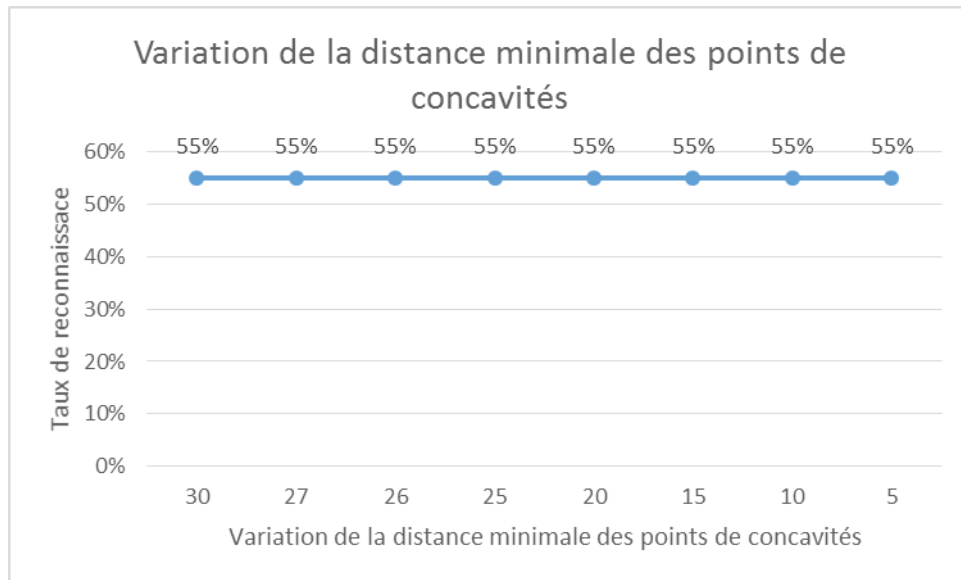


FIGURE 31 – Taux de reconnaissance par rapport à la variation de distance minimale des points de concavité.

Détermination de la distance minimale des points de concavité On remarque immédiatement, que le taux de reconnaissance est inchangé dans la variation de la distance minimale des points de concavité (Figure 31).

Ce résultat est expliqué par le fait que le calcul du résultat du classifieur est principalement basé sur le nombre de points convexes et non celui du nombre de points concaves. A partir de là, nous partons du principe que les deux distances peuvent prendre la même valeur : $\epsilon = \beta = 25$.

Détermination de la distance minimale entre points de convexités

On remarque que le taux de reconnaissance s'accroît au fur et à mesure que la distance minimale diminue jusqu'à atteindre un maximum de 55% à partir de la valeur 15.

Ce résultat est expliqué par le fait que cette distance permet de détecter les doigts levés consécutifs de la main avec une distance moyenne entre 15 et 5. Cependant, cette distance ne permet pas de détecter efficacement les doigts levés éloignés comme pour faire des signes avec deux doigts éloignés. Par contre, elle permet de détecter efficacement les combinaisons de doigts au dessus de 2. De cette série de test, on déduit que la valeur intéressante de μ est 15, car elle permet de d'obtenir à la fois le meilleur taux mais aussi potentiellement une détection plus efficace pour les signes de la main contenant des combinaisons de doigts inférieurs à 2.

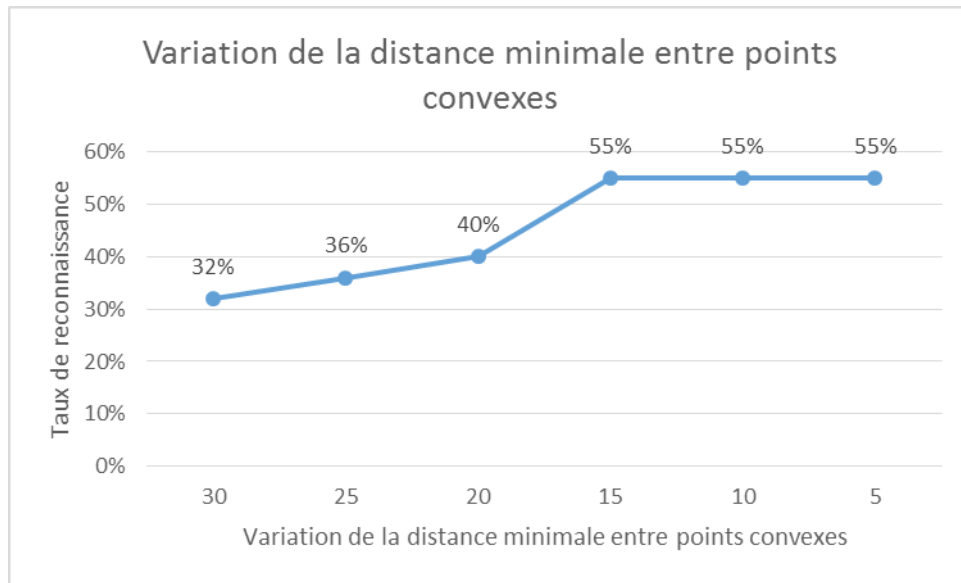


FIGURE 32 – Taux de reconnaissance par rapport à la variation de distance minimale entre points convexes.

4.4.4 Conclusion

Avec les séries de test effectuées, on a pu atteindre un maximum de 55% de taux de reconnaissance sur la base d'apprentissage sur laquelle l'ensemble des classifieurs s'est basé. De plus, ce classifieur ne nécessite pas de phase d'apprentissage. Il permet des calculs plus rapides et donc permettra à terme de pouvoir trancher les ambiguïtés lors de la combinaison de tous les classifieurs.

4.4.5 Remarques

Cependant, cet algorithme ne nous permet pas de dépasser un taux de reconnaissance au dessus de 55%, car il a été admis sur des conditions d'acquisition restreintes, telles que :

- La détection d'une seule main à la fois
- La main doit être à une distance constante par rapport à la caméra
- La main doit être levée perpendiculairement au bras
- Les doigts de la main doivent pointer (dans l'idéal vers le haut)
- Les signes de la main doivent avoir uniquement les doigts levés de manière successive
- Lors de la segmentation, seule la main est sauvegardée (en dehors du poignet et tout autre forme parasite lors de l'acquisition).

5 Combinaison des classifieurs

5.1 Description de la méthode

Enfin, nous avons combiné nos différents classifieurs, par produit des probabilités obtenues à travers ces différents classifieurs. Dans le cas du classifieur par détection de convexité, le résultat obtenu n'est pas un vecteur de probabilité associée à chaque classe, mais un nombre de doigt. Afin de combiner ce résultat aux vecteurs de probabilité retournés par chaque autre classifieur, nous avons défini la probabilité de la classe retournée à 1, toutes les autres étant mises à 0. Nous avons combiné les résultats par somme des vecteurs de probabilité d'appartenance à chaque classe, en effet combiner par produit donnerait toujours le résultat du classifieur par convexités.

5.2 Analyse des résultats

Seul le classifieur par histogramme radial utilise la méthode des KPPV dans les tests effectués, ce qui lui donne une prépondérance lorsque $k < NB_CLASSES$ sur les classifieurs par profil, par histogramme, et par zoning.

Ainsi, les différents vecteurs de probabilité ont été combinés par produit. Nous avons testé une base d'apprentissage de 49 images au total. Ce sont par ailleurs ces mêmes images que nous avons testé. Bien que cela puisse fausser certains résultats, notamment dans le cas où K vaut 1 (pour le classifieur par histogramme radial), nous avons considéré que notre base était suffisamment étendue pour que l'erreur induite reste assez faible.

Finalement, nous obtenons un taux de reconnaissance compris entre 85.7% et 89.8% de réussite dans le cas où $K = 1$, en fonction des différents paramètres des autres classifieurs (nombre de lignes à considérer dans le cas des classifieurs par profil et par histogramme, nombre de subdivisions de l'image dans le cadre du classifieur par densité), nous obtenons entre 51% et 53% lorsque K vaut 2, enfin à partir de $K = 3$ nos résultats se stabilisent à 44.9%.

Afin d'obtenir des résultats plus probants, il nous faudrait disposer d'une base d'apprentissage plus étendue, mais également d'un jeu d'images test non présentes dans la base d'apprentissage. Par ailleurs, dans les tests effectués nous n'avons pas utilisé la méthode des KPPV pour les classifieurs par profil, par histogramme et par densité, ce qui rend leur influence secondaire par rapport au classifieur par détection de convexité (dont les probabilités retournés sont binaires, 0 ou 1), mais également par rapport au classifieur par histogramme radial dans le cas où $K < NB_CLASSES$. Ainsi, leur influence au cours de ces tests reste modeste, et ce quelques soient les paramètres associés à ces trois classifieurs.

6 Suivi de la main dans une séquence d'images

6.1 Présentation de la méthode

Les images du groupe 2, qui constituent une succession d'images, ne peuvent être traitées de la même manière que celles des deux autres groupes. En effet, notre méthode de segmentation telle qu'elle est conçue est idéale sur des images dans lesquelles la main est en avant-plan, ce qui n'est pas le cas sur l'ensemble d'images de ce groupe. Nous avons donc développé une méthode de traitement spéciale pour ce groupe, effectuant un suivi de la main dans l'image. L'idée générale de notre méthode est de suivre la main d'une image à l'autre afin de seuiller et ne garder uniquement la partie de l'image qui contient la main. Un tel algorithme pourrait être utilisé pour réaliser du suivi de main dans une séquence d'images quelconque, indépendamment de la reconnaissance. Notre méthode nécessite cependant de bénéficier à la base d'une image propre sur laquelle il est possible de réaliser une segmentation idéale. L'algorithme de suivi de la main est basé sur l'opérateur de Harris. Il permet d'effectuer un seuillage localisé sur la main d'une image à l'autre. Afin de localiser la main, on part à la base d'une image seuillée et on calcule la boîte englobante de la main. Cette même boîte englobante est utilisée sur l'image suivante de sorte que l'on obtienne deux fenêtres d'intérêt pour chaque image, partant du postulat que d'une image à l'autre, le mouvement de la main est minime. L'opérateur de Harris est alors appliqué sur chaque fenêtre (chaque main) de deux images consécutives. Après un appariement des points, on peut calculer un mouvement de la main par distance euclidienne de deux points appariés. Une nouvelle boîte sur laquelle on applique le mouvement est alors calculée et appliquée à la seconde image. Cette méthode est théoriquement efficace pour des mouvements latéraux de la main, mais lorsque celle-ci se rapproche ou s'éloigne de la caméra, un ajustement est nécessaire. C'est la raison pour laquelle la taille d'une boîte englobante est élargie avant seuillage puis réadaptée.

L'algorithme développé peut être écrit comme suit :

- Segmentation de la main sur la première image.
- Création de la boîte englobante (box).
- Opérateur de Harris pour identifier les points d'intérêts.
- Boucle : Pour chaque image consécutive
 - Élargissement de la box.
 - Seuillage à l'intérieur de la box afin d'extraire la main.
 - Recadrage de la box si nécessaire.
 - Opérateur de Harris pour identifier les points d'intérêts.
 - Appariement des points de l'image seuillée et de la précédente.
 - Application du mouvement de la main sur le déplacement de la box.

6.1.1 Segmentation de la main et extraction de la box

Cette segmentation est « banale » et réutilise la méthode de segmentation générale basée sur l'algorithme de Fisher. A partir de l'image binarisée

résultante, il est très simple de calculer les coordonnées de la boite englobante de la main extraite (Figure 33).

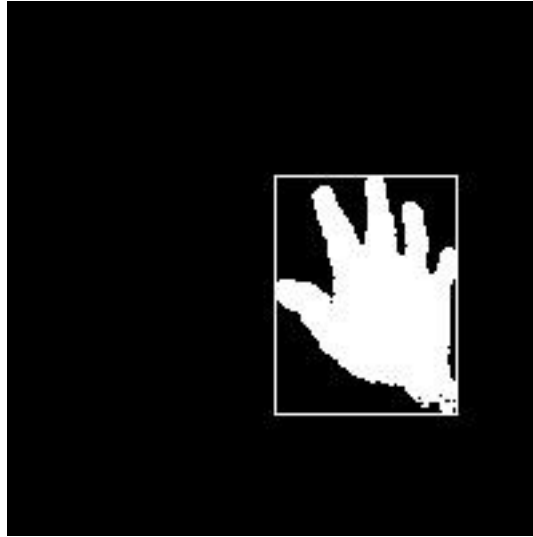


FIGURE 33 – Boîte englobante d’une main extraite.

6.1.2 Opérateur de Harris

Dans la seconde étape importante, l’opérateur de Harris a été utilisé afin d’extraire les points d’intérêt de l’image seuillée. Ces points d’intérêt auraient pu être calculés sur l’image source, mais d’une image à l’autre, l’expérience a prouvé qu’aucuns points ne correspondaient, ce qui s’avérait problématique pour réaliser l’appariement d’une image à l’autre et détecter un mouvement. Sur l’image suivante, on observe les contours de la main ainsi que les points d’intérêt en blanc (Figure 34).

Pour chaque image consécutive :

6.1.3 Seuillage sur la boite englobante

Afin de préparer le terrain à l’opérateur de Harris, il est nécessaire de seuiller chaque image. L’identification du seuil est l’étape de l’algorithme qui présente le plus de problèmes et également celle qui est le plus à même d’être améliorée. Explications : Selon l’image de la séquence qui est traitée, en partant du principe que la fenêtre est toujours centrée sur la main, il est évident que le contraste sera plus ou moins intense, que la profondeur de la main sera plus ou moins élevée, et donc que l’histogramme de cette main sera toujours différent d’une image à l’autre. La méthode de seuillage utilisée doit donc ici être particulièrement bien choisie si l’on veut que qu’elle fonctionne pour toutes les images d’une séquence.

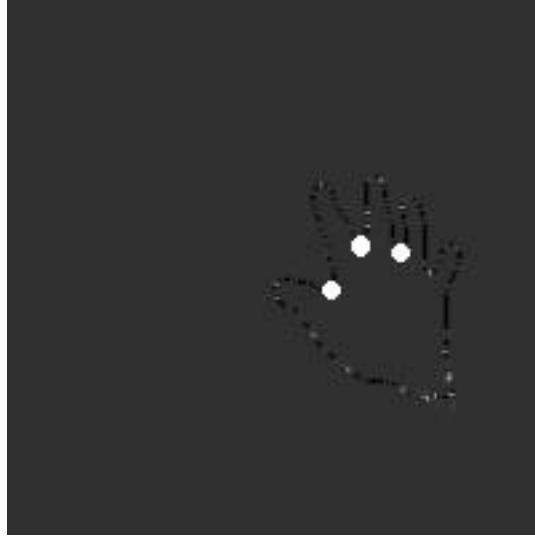


FIGURE 34 – Points d'intérêts de la main par l'opérateur de Harris.

Exemple de deux histogrammes selon une luminosité et un contraste différent : (Figure 35).

Comme l'atteste le tableau ci-dessus (Figure 35), nous avons affaire à des histogrammes bimodaux, même si c'est moins prononcé dans le second exemple de la main claire (en raison du nombre très élevé de valeurs de blanc). Il est cependant possible que l'histogramme ne suive pas le même schéma et notamment que le deuxième mode ne soit pas 255 par exemple si l'objet se trouve derrière la main ou si la main est devant le corps. La technique utilisée consiste à couper l'histogramme en deux parties via sa médiane (non pas sa moyenne). On recherche ensuite le mode sur le premier sous histogramme qui représente le niveau de gris le plus présent dans la main. On retient pour valeur de seuil un niveau de gris un peu plus élevé que le mode afin de seuiller la majeure partie de la main.

$$Thresh = mode + n$$

Ce paramètre n égal à 10 doit varier selon le contraste de la main. En effet, ce paramètre dépend de l'étendue de la première cloche de l'histogramme représentant le contraste de la main. Un faible contraste demandera une valeur faible tandis qu'un grand contraste demandera une valeur plus élevée. Tout l'enjeu ici est de seuiller correctement la main. Si le seuillage est trop faible ou trop élevé la binarisation donnera une image intraitable par le système de reconnaissance.

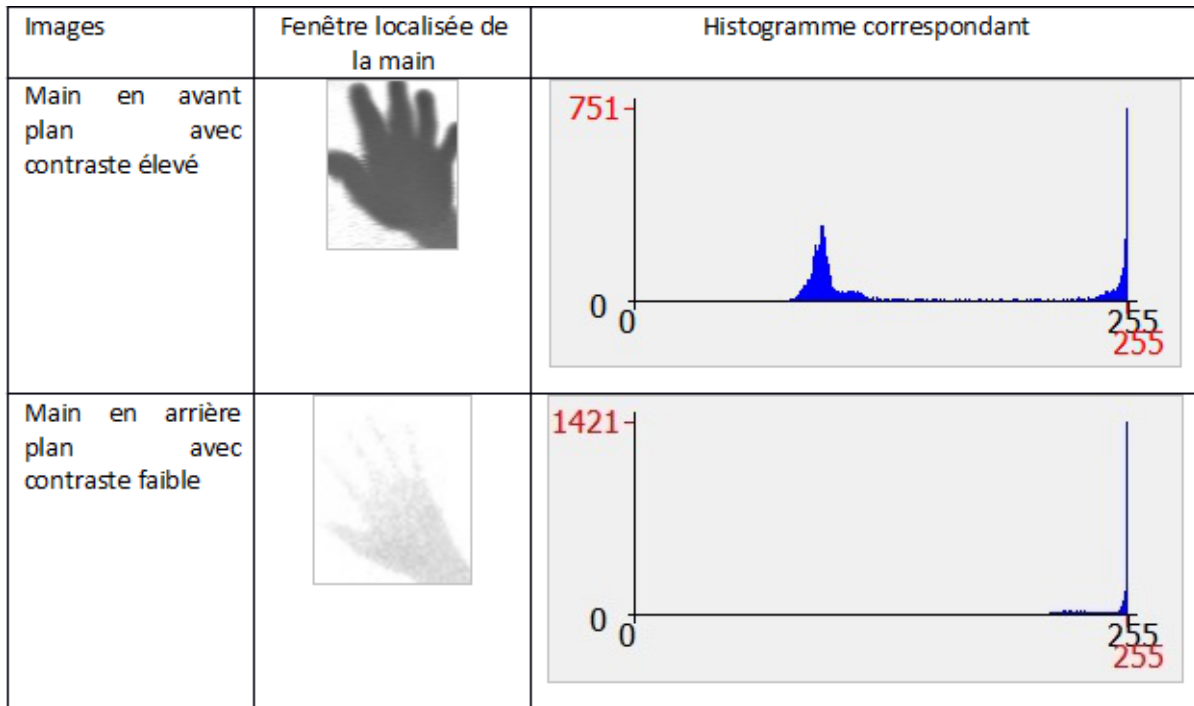


FIGURE 35 – Histogrammes selon la luminosité et le contraste.

6.1.4 Harris + Appariement des points de deux images seuillées

Une fois le seuillage et la segmentation effectués, il est important de détecter le mouvement de la main afin d'obtenir une nouvelle box pour cette seconde image. Si l'on se contentait de conserver l'ancienne box dont on s'est servi pour segmenter la nouvelle image, on pourrait d'une image à l'autre perdre de l'information dans la mesure où la main se déplace.

Solution développée : On applique l'opérateur de Harris sur la nouvelle image seuillée afin de réaliser l'appariement avec l'image précédente, le but étant d'identifier le mouvement, déplacement des points d'intérêt.

Une première solution a été implémentée afin d'apparier les points. D'une image à l'autre, les points d'intérêt étaient rattachés dans la mesure où leur distance euclidienne (de leurs coordonnées spatiales) était minimale. Cette méthode présentait le désavantage de chercher à minimiser le mouvement d'une image à une autre, ce qui ne correspondait pas forcément à la réalité.

Une autre méthode de mise en corrélation a été développée et présente des résultats bien plus satisfaisants : SAD (Sum of Absoute Differences)

Pour chaque point d'intérêt, on calcule la mesure de similarité entre une fenêtre de taille 5 (paramètre variable) et la fenêtre du point d'intérêt d'une autre image :

$$\sum_{(u,v) \in W} |I_1(u,v) - I_2(u,v)|$$

Considérant : W la fenêtre d'encadrement d'un point d'intérêt.

La mesure de similarité est calculée sur l'image source et non l'image seuillée pour de meilleurs résultats. Une fois la valeur de similarité calculée entre chaque point d'intérêt des deux images, il faut les rattacher. Pour cela, on les rattache par leurs valeurs minimales. Un test est effectué sur la distance euclidienne entre les deux points afin d'éviter de rattacher des points trop éloignés. Cette étape évite de générer des outliers qu'il faudrait alors éliminer. Un point ne peut être apparié deux fois. Exemple d'appariement pour deux images consécutives : (Figure 36).

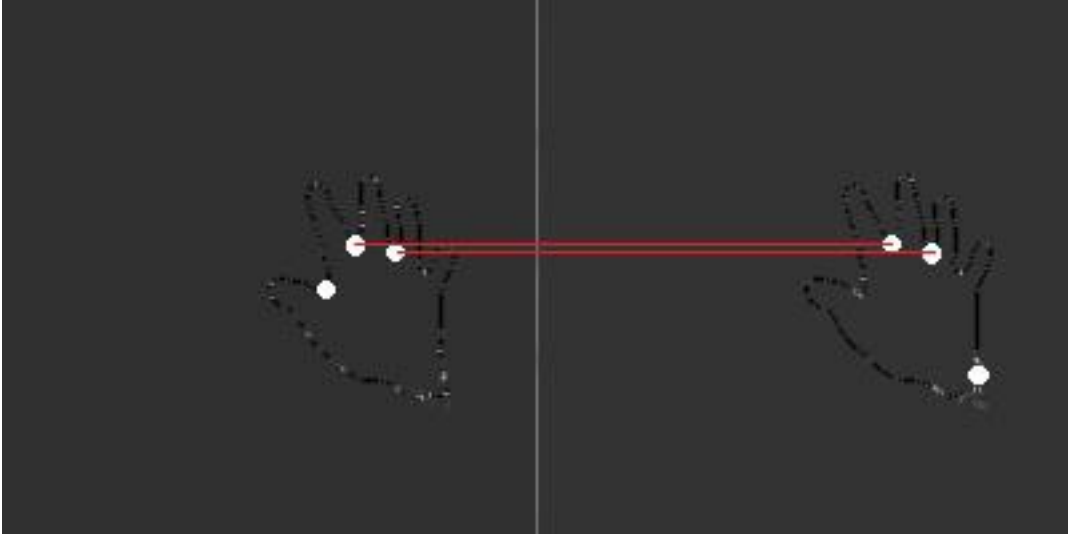


FIGURE 36 – Appariement pour deux images successives.

La mise en correspondance des points permet d'obtenir un vecteur de translation (a, b) pour chaque couple de points mis en corrélation. On peut alors obtenir un vecteur de translation moyen d'une image à l'autre : (A, B) .

6.1.5 Déplacement de la box

Une fois obtenue l'estimation d'un mouvement d'une image à l'autre, on peut alors appliquer le vecteur de translation aux coordonnées de la boîte englobante sur la seconde image. Afin d'avoir un encadrement correct de la main. Ainsi, même si l'on perd un peu d'information sur le seuillage appliqué à la boîte englobante d'une ancienne image, on le corrige grâce au mouvement obtenue par cette méthode, ce qui nous permet de conserver un encadrement correct de la main.

6.1.6 Redimensionnement de la boîte

Il y a un dernier point qui pose cependant problème et auquel nous avons du répondre : Lorsque la main est en mouvement, elle peut s'approcher ou s'éloigner de la caméra, ce qui est l'enjeu principal de la séquence d'images. De ce fait naît une problématique : Sans considérer le seuillage de la main, celle-ci s'agrandit ou se rétrécit, et la taille de la boîte englobante doit donc être modifiée en conséquence.

Solution envisagée :

- A chaque boucle (traitement d'une nouvelle image), la boîte est élargie d'un pixel dans toutes les directions avant le seuillage.
- La boîte est redimensionnée si nécessaire après seuillage et segmentation de l'image.

6.2 Analyse des résultats

Si la méthode générale développée et expliquée ci-dessus ne donne que des résultats moyennement satisfaisants, elle en donne tout de même puisque nous arrivons à effectuer un suivi de la main, au moins un temps. On remarque que le suivi de la main dépend énormément du type de seuillage effectué. De nombreux tests donnant des séquences de résultats différents ont été effectués afin de développer une méthode de seuillage la plus efficace possible. Actuellement, la méthode de découpage d'histogrammes donne les « meilleurs » résultats, ou permet tout du moins de se focaliser sur la main, même si celle-ci est très rapidement mal seuillée et que la boîte englobante n'englobe pas toute la main. (cf vidéo `methode2.0002.wmv`). Une autre méthode de seuillage a été développée et présente des résultats différents : (cf vidéo : `methode1.0001.wmv`). Cette méthode basique consiste à prendre le Xème niveau de gris non nul de la fenêtre encadrant la main. Elle présente un résultat différent puisqu'il est également efficace au début et rencontre une difficulté lorsque le poignet apparaît nettement dans l'image.

La raison pour laquelle ces deux méthodes ne fonctionnent pas convenablement est double :

Tout d'abord, le seuillage local n'est pas optimal car pas très intelligent. Compte tenu de l'impact du seuillage sur la qualité de la segmentation, une première amélioration de la détection de la main non négligeable consisterait à parfaire ce seuillage.

- Des tests ont été tentés à l'aide de l'algorithme de Fisher, mais en raison de difficultés techniques et de manque de temps, nous n'avons pu approfondir cette méthode qui devrait néanmoins pouvoir donner des résultats satisfaisants, découpant un histogramme en N classes.
- Une segmentation par approche régionale (quadtree, graphes, ...) pourrait être tentée et peut-être apportée des résultats corrects.

La seconde partie du problème réside dans l'adaptation de la box et notamment dans la phase d'agrandissement. Dans la séquence d'images étudiée, lorsque la main passe en arrière plan, le poignet se trouve confondu avec la main

et est donc considéré comme en faisant partie (Méthode 1). A l'inverse, avec la seconde méthode de seuillage, la box se rétrécit pour n'encadrer que la paume de la main.

On comprend donc que l'adaptation de la box, si elle est indispensable va dépendre de la méthode de seuillage utilisée et donner une segmentation complètement différente suivant les techniques. Une évolution du programme est donc de trouver le seuillage adapté à chaque type d'image.

6.3 Améliorations possibles

- Développement et test de nouvelles méthodes de seuillage.
- Développer d'autres méthodes de corrélation pour Harris afin d'affiner la détection de mouvement.
- Dans la suivi de mouvement, lorsque l'on traite deux images, une fois la nouvelle boîte englobante calculée pour la deuxième image, il serait envisageable de resegmenter cette image afin d'être sûr de ne pas avoir perdu d'information. Cela devrait théoriquement éviter d'avoir parfois des mains rognées et donc améliorer la reconnaissance ensuite.

Conclusion

Ce projet de traitement d'image nous a permis de mettre en œuvre les connaissances acquises durant ce semestre. Il nous a permis également de concevoir et développer un système de reconnaissance d'image depuis la base d'acquisition jusqu'à la phase décisionnelle associée au système de reconnaissance.

Après avoir effectué un état de l'art, nous avons eu loisir de nous confronter aux problématiques de segmentation d'images et de reconnaissances de forme en développant différents classifieurs combinés dans un système de reconnaissance de signes de la main. Chaque phase de développement associée à une étape du système de reconnaissance, comme la segmentation, la classification ou encore l'algorithme de suivi de la main dans une séquence d'images, ont tous sollicité la théorie émise durant les cours ainsi que notre intérêt à relever un défi. Si notre projet présente des résultats discutables, notre implication dans ce dernier elle ne l'est pas, et nous apprécions le fait d'avoir pu apprendre à développer au sein des UV IN52 et IN54 des méthodes de traitement de l'image.

Nous n'avons malheureusement pas eu le temps d'associer la reconnaissance au suivi de la main dans une séquence image, qui n'aurait de toute manière pas donné de résultats probants. Cependant, une évolution possible de notre programme serait de récupérer le flux de la caméra et de réaliser une reconnaissance en temps réel des signes de la main réalisés par un utilisateur filmé. Nous avons conduit notre projet en considérant cet objectif comme une fin.

Nous sommes par ailleurs également satisfaits d'avoir pu apprendre à utiliser la bibliothèque de traitement d'images en temps réel : OpenCV.

Références

- [1] « Recognizing Hand Gestures with Microsoft's Kinect », Matthew Tang, Department of Electrical Engineering, Stanford University
http://www.stanford.edu/class/ee368/Project_11/Reports/Tang_Hand_Gesture_Recognition.pdf accédé pour la dernière fois le 6/01/2013
.
- [2] « Hand Gesture Recognition Using Kinect », Heng Du et TszHang To, Université de Boston
<http://iss.bu.edu/data/jkonrad/reports/HDTT11-04buece.pdf>
accédé pour la dernière fois le 7/01/2013.
- [3] Segmentation des images, Urfist Paris
http://urfist.enc.sorbonne.fr/anciensite/image_numerique/segmentation.htm accédé pour la dernière fois le 7/01/2013.
- [4] Otsu Thresholding, The Lab Book Pages
<http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html> accédé pour la dernière fois le 7/01/2013.