

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

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

7 janvier 2013

Table des matières

1	Présentation globale du programme	2
2	Segmentation	2
2.1	Présentation de la méthode	2
2.2	Analyse des résultats	6
3	Classification	12
3.1	Classifieurs par profils et histogrammes horizontaux et verticaux	12
3.1.1	Présentation de la méthode	12
3.1.2	Résultats	14
3.1.3	Améliorations possibles	14
3.2	Classifieur densités	16
3.2.1	Résultats	16
3.3	Classifieur histogramme radial	17
3.3.1	Présentation de la méthode	17
3.4	Classifieur de convexité	18
3.4.1	Présentation	18
3.4.2	Tests : paramètres optimaux	22
3.4.3	Conclusions	25
3.4.4	Remarques	25
4	Combinaison des classifieurs	25
5	Suivi de la main dans une séquence d'images	25
5.1	Présentation de la méthode	25
5.1.1	Segmentation de la main et extraction de la box	26
5.1.2	Opérateur de Harris	27
5.1.3	Seuillage sur la boîte englobante	27

Introduction

1 Présentation globale 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 compatible 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.

Considérant ce résultat peu satisfaisant (Figure 1), 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 2), 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 3). 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 1 – Résultat de l'interpolation linéaire des valeurs flottantes de l'image source.



FIGURE 2 – Résultat de l'interpolation basée sur la moyenne des valeurs.

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-



FIGURE 3 – Résultat de la conversion quadratique.

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



FIGURE 4 – 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 5).



FIGURE 5 – 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 6).

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 de grignoter le 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 7).

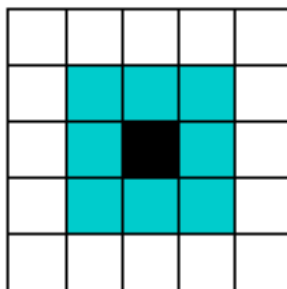


FIGURE 6 – Pixel connexité 8.

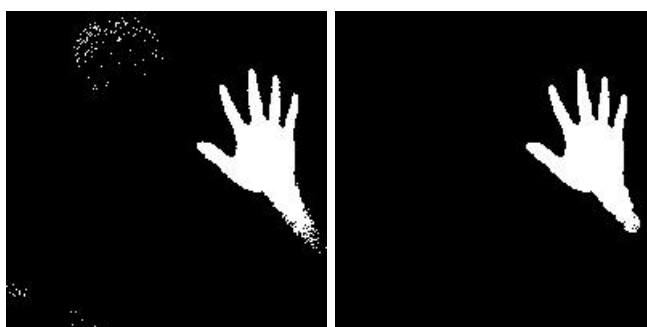
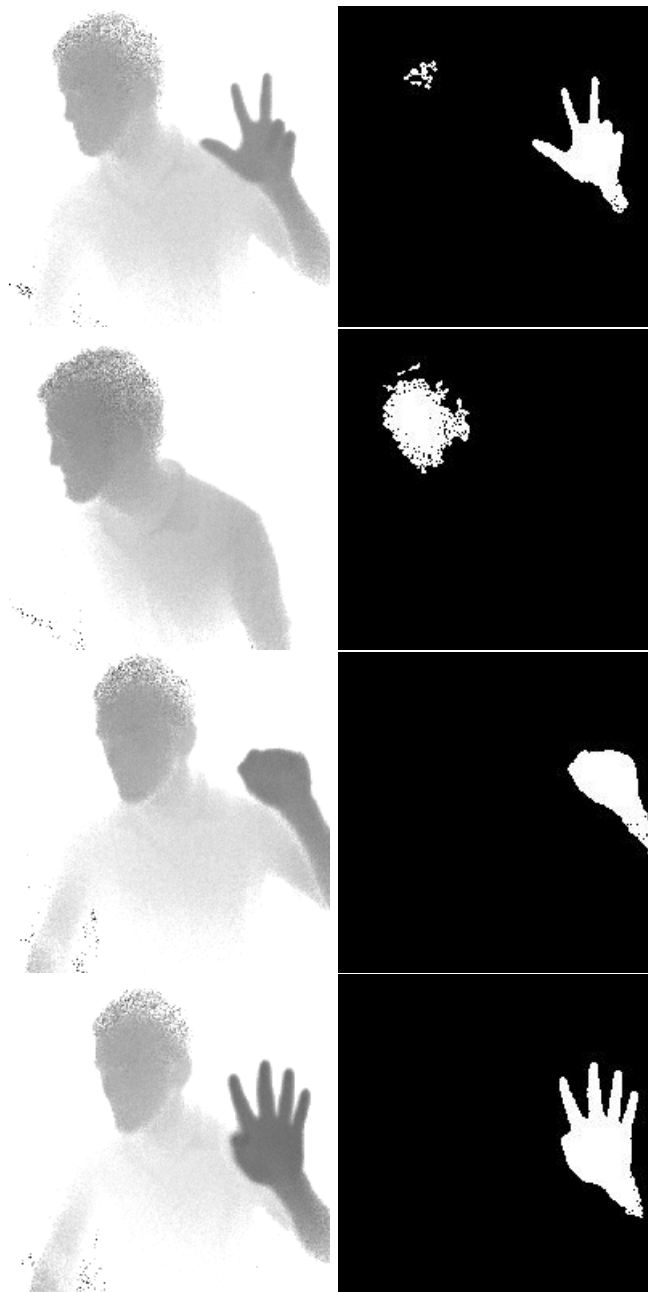


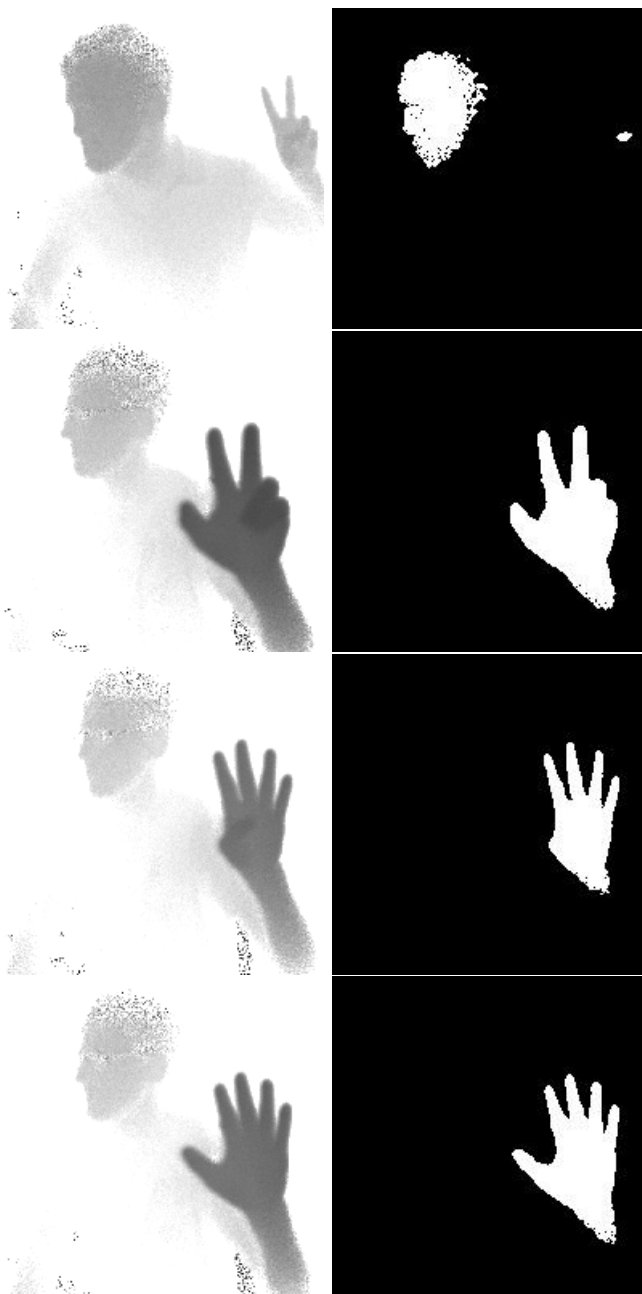
FIGURE 7 – Résultat de la segmentation pour la première image du groupe 1, avant et après filtrage.

2.2 Analyse des résultats

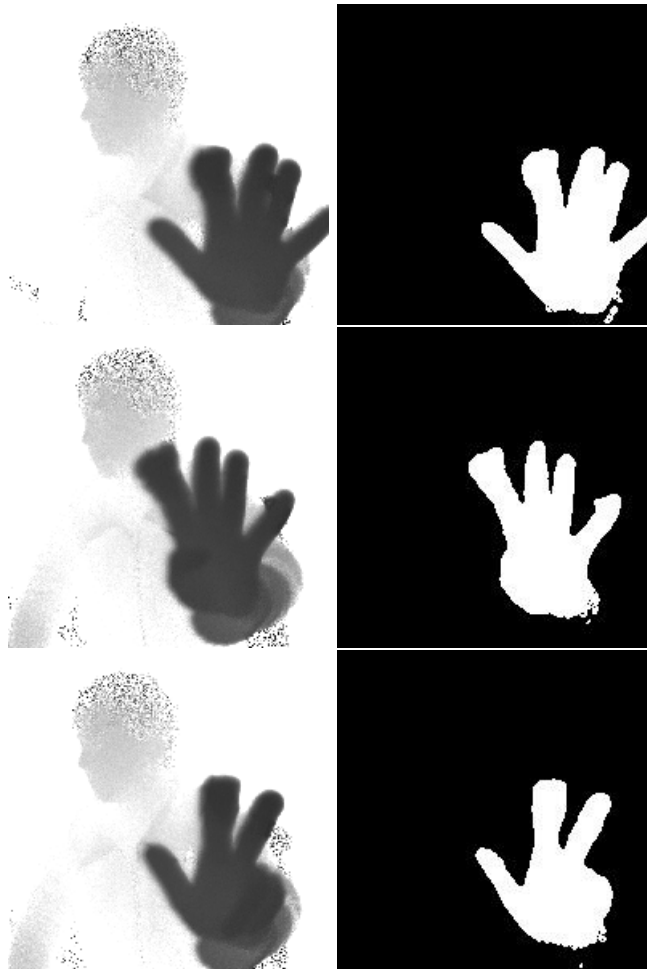
Cette méthode de segmentation se trouve être efficace sur la majorité des images fournies dans la base. Voici un panel de test réalisé sur l'ensemble des images du groupe 1 :











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 (3^{eme} image)
- Lorsque la main n'est pas en avant-plan (images 2 et 6).

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





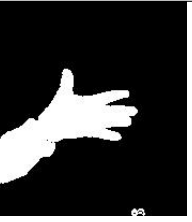


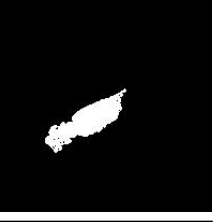




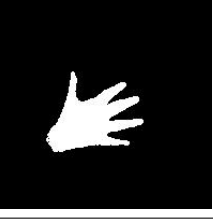


Image source	Image non corrigée avant segmentation		Image corrigée avant segmentation	
				
				
				

FIGURE 8 – Résultats de la correction.

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). 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.

3 Classification

3.1 Classifieurs par profils et histogrammes horizontaux et verticaux

3.1.1 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 9 – Profils verticaux d'une main.



FIGURE 10 – 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. A 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 ligne 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.

3.1.2 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 moyennement autour 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 configurations 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 11).

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 12).

3.1.3 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

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

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

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.

3.2 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 obtiens ainsi la densité des différentes zones de l'image.

Sur le même principe que pour les classifieurs précédents, on établie 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 subdivision de l'image est également paramétrable via les variables M et N. A nouveau, ce nombre de zone 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 satisfaisant. A l'inverse, un trop grand nombre de zones peut produire des écarts importants, et le taux de reconnaissance se dégrade.

3.2.1 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 13 – 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 14 – Résultats du classifieur densités avec KPPV.

3.3 Classifieur histogramme radial

3.3.1 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. 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.

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 :

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

FIGURE 15 – Exemple d'une image de main segmentée et binarisée.

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 16 – Image de décalages d'angles obtenue à partir de (Figure 15)

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 17).

FIGURE 17 – Histogramme radial obtenu à partir de (Figure 15)

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 (Figure 15). C'est pourquoi nous utilisons à la place du centre de gravité de la main le centre de la paume de la main, que nous calculons en supprimant les doigts de l'image binarisée par érosion avec un élément structurant suffisamment grand pour supprimer entièrement les doigts de la main. Celui-ci étant placé plus

FIGURE 18 – Image segmentée et binarisée d’une main, avec en vert le centre de gravité de la main et en rouge le centre de gravité de la main.

FIGURE 19 – Histogramme radial avec centre de la paume obtenu à partir de (Figure 15)

bas sur la main, celui-ci produit des pics plus distincts pour chaque doigt dans l’histogramme radial (Figure 19).

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 :

3.4 Classifieur de convexité

3.4.1 Présentation

Ce classifieur est basé sur le rapport « Hand Gesture Recognition Using Kinect » de Heng Du et TszHang To de l’Université de Boston. 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.

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 que l’image de la main ait été extraite de l’image d’origine puis segmentée, on a utilisé la fonction implémentée d’OpenCV `findContours()` et pouvoir ainsi dérouler l’algorithme uniquement sur le contour de la main. Comme illustrée par la figure suivante : (Figure 20).

Approximation de la forme de la main par un polygone L’étape suivante permet de simplifier la détection des points de convexités et de concavités. Pour cela, on utilise la fonction implémentée d’OpenCV `approxPolyDP()` basé sur l’algorithme récursif de Ramer-Douglas-Peucker qui réduit le nombres de



FIGURE 20 – Contour de la main par findContours.

points d'une courbe en approximant une courbe similaire avec 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 éloignée 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 réapplique l'algorithme récursivement sur les deux segments : (premier point, nouveau point) et (nouveau point, dernier point).
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 convexités 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 21).

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 concavités en utilisant une fonction adaptée d'OpenCV `cvConvexityDefects()`. Comme illustrée par la figure suivante : (Figure 22).



FIGURE 21 – Polygone englobant de la main tel que claculé par `convexHull`.

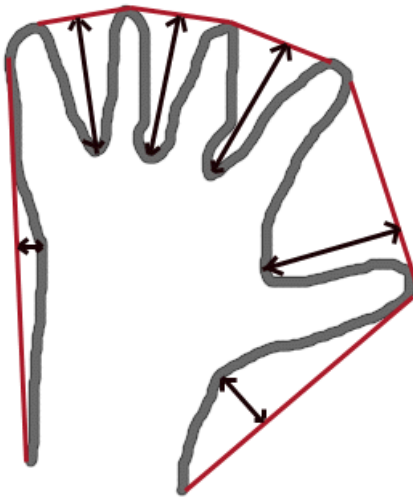


FIGURE 22 – Points de concavité tels que détectés par `cvConvexityDefects`.

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 paramètres statiques prédéterminés avant le lancement de cette dernière. Tout d'abord en calcul un rectangle approximative de la paume de la main en utilisant la fonction implémentée d'OpenCV `minAreaRect()`. Par la suite, afin de filtrer les points de convexités et respectivement de concavités, on compare

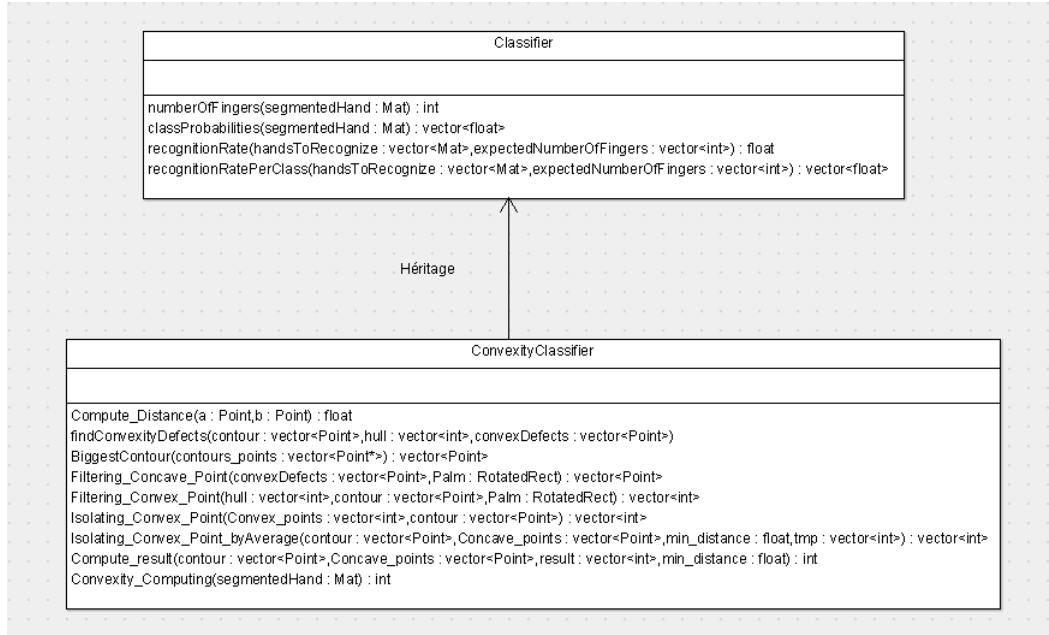


FIGURE 23 – Diagramme UML du classifieur par convexités.

leurs coordonnées en y avec le centre approximative de la paume + ϵ (> 0 valeur donnée) et respectivement β . Ces dernières nous permettent d'affiner nos résultats en augmentant la coordonnée en y du centre approximative de la paume et ainsi réduire le nombre de points gardées.

Puis une fois les points de concavités et de convexités filtrés, on filtre une dernière fois les points de convexités selon leurs distance 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

3.4.2 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 optimum pour les 3 paramètres optimaux :

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

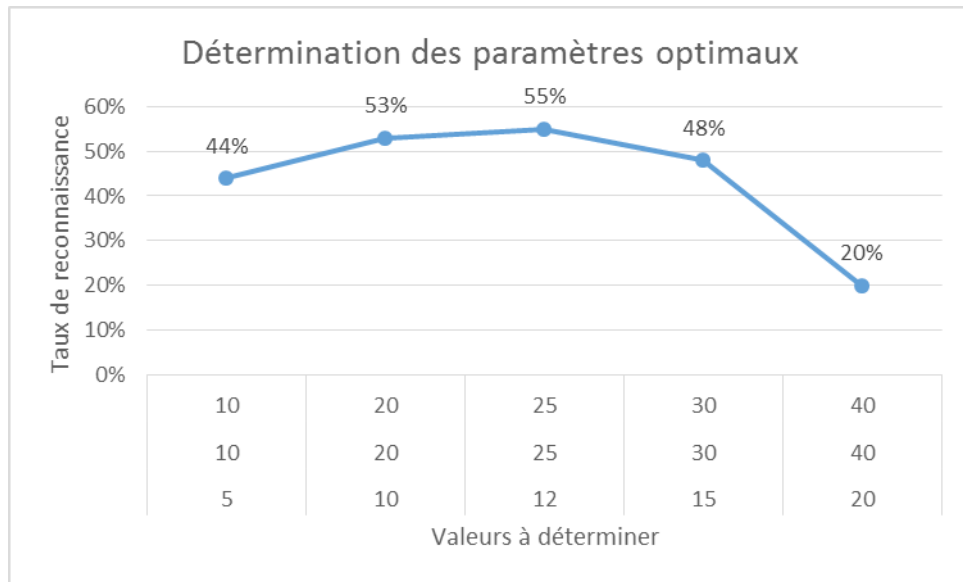


FIGURE 24 – Taux de reconnaissance par rapport aux paramètres ϵ , β et μ .

On remarque que le taux de reconnaissance atteint son maximum sur une plage de valeurs de [25, 25, 12] (Figure 24).

A partir de ces tests préliminaires, on a peu réduire les intervalles de valeurs pour la détermination des paramètres individuellement.

Détermination de la distance minimale des points de convexités On remarque que le taux de reconnaissance s'améliore dans une fourchette de valeur entre [27 , 20] et redescend en flèche en dehors de cet intervalle (Figure 25).

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 convexités, ce qui expliquera ainsi cet amélioration du taux de reconnaissance. En dessous, les points de convexités à considérer seront trop importants et au dessus le nombre de points de convexités seront insuffisant pour pouvoir déterminer avec précision le résultat attendu. On obtient alors une distance optimale de $\epsilon = 25$.

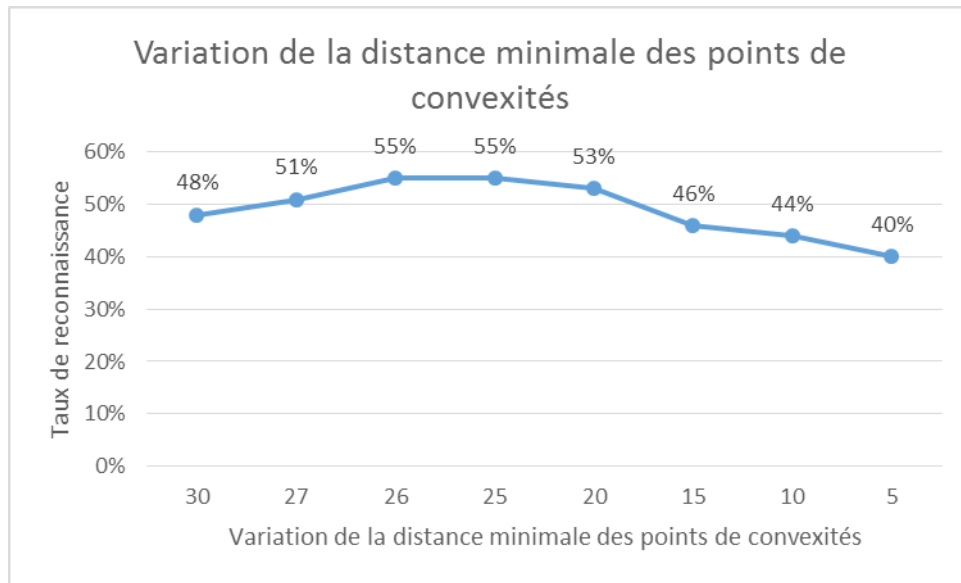


FIGURE 25 – Taux de reconnaissance par rapport à la variation de distance minimale des points de convexités.

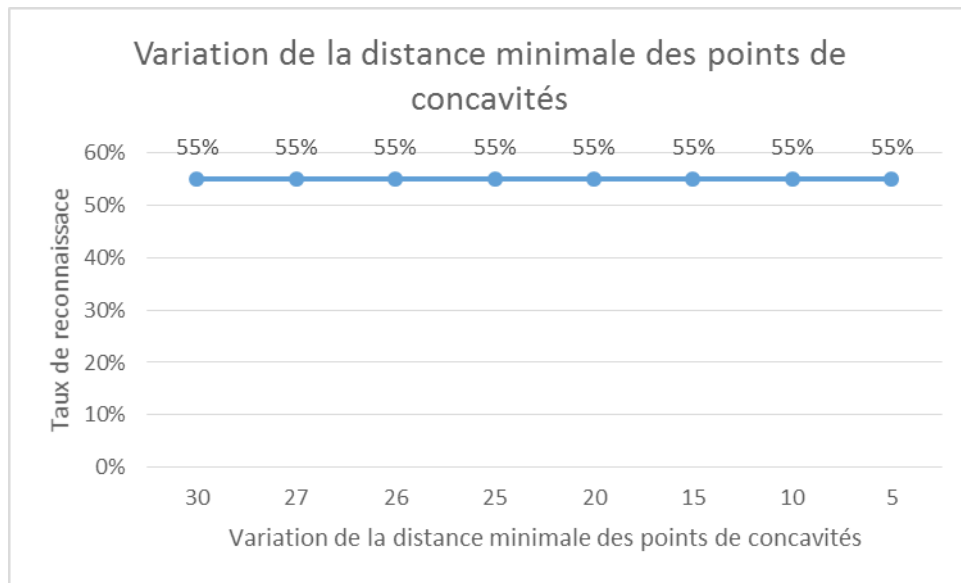


FIGURE 26 – Taux de reconnaissance par rapport à la variation de distance minimale des points de concavités.

Détermination de la distance minimale des points de concavités On remarque immédiatement, que le taux de reconnaissance est inchangé dans la variation de la distance minimale des points de concavités (Figure 26).

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 nombres de points concaves. A partir de là, on part du simple principe les deux distances peuvent prendre la même valeur : $\epsilon = \beta = 25$.

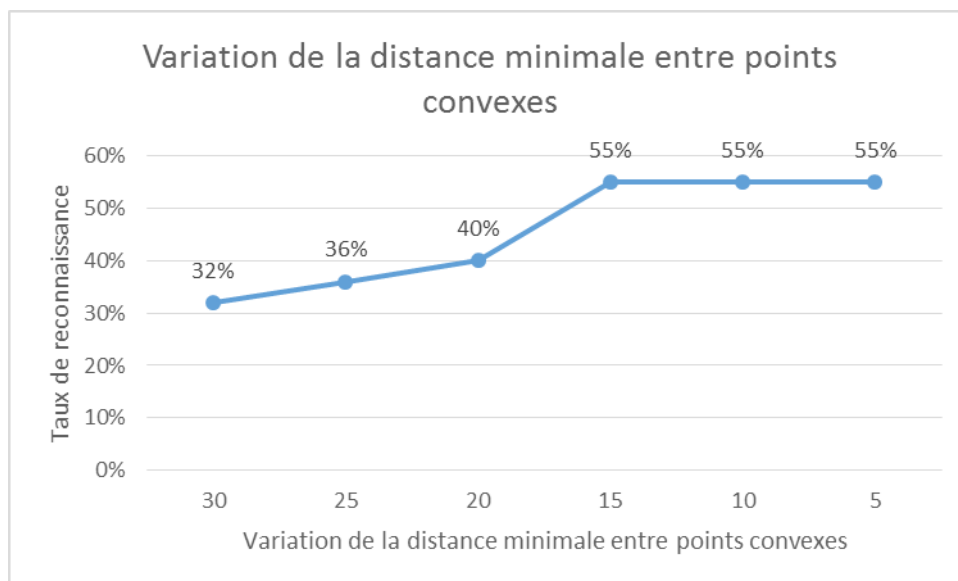


FIGURE 27 – Taux de reconnaissance par rapport à la variation de distance minimale entre points convexes.

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 faite que cette distance permet de détecter les doigts levés consécutives 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 doigts au dessus de 2. De cette série de test, on en 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.

3.4.3 Conclusions

Avec les série de test effectuées, on a pu atteindre un maximum de 55% de taux de reconnaissance sur la base d'apprentissage sur lesquels nos classifieurs se sont tous basées. De plus, ce classifieur ne nécessitant pas de phase d'apprentissage il permet des calculs plus rapides et donc permettra à terme de pouvoir trancher lors des ambiguïtés lors de la combinaison de tous les classifieurs.

3.4.4 Remarques

Cependant, cet algorithme ne nous permet pas de dépasser un taux de reconnaissance au dessus de 55%, car l'algorithme sur lequel ce dernier est basé a été admis avec des conditions d'acquisitions restreintes, tel 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 doit pointé (dans l'idéale vers le haut)
- Les signes de la main doit avoir uniquement les doigts levés dand l'ordre successives.
- Lors de la segmentation seul la main est sauvegarder (en dehors du poigné et tout autre formes parasites lors de l'acquisition).

4 Combinaison des classifieurs

5 Suivi de la main dans une séquence d'images

5.1 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 seillonner 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
 - Elargissement 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.

5.1.1 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 binaire résultante, il est très simple de calculer les coordonnées de la boîte englobante de la main extraite (Figure 28).

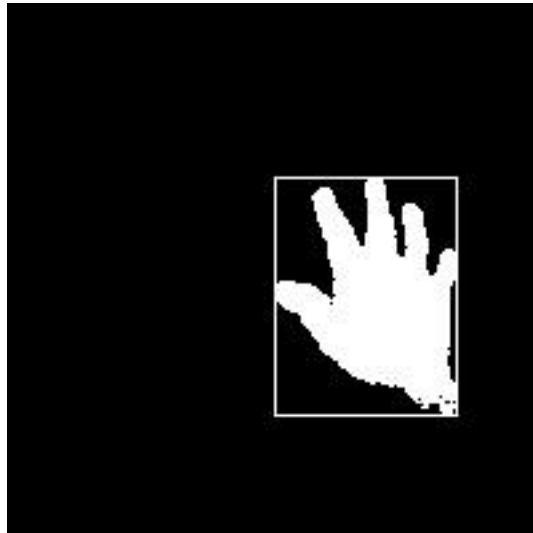


FIGURE 28 – Boîte englobante d'une main extraite.

5.1.2 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 était donc problématique pour réaliser l'appariement d'une image à l'autre et de détecter un mouvement. Sur l'image suivante, on observe les contours de la main ainsi que les points d'intérêt en blanc (Figure 29).

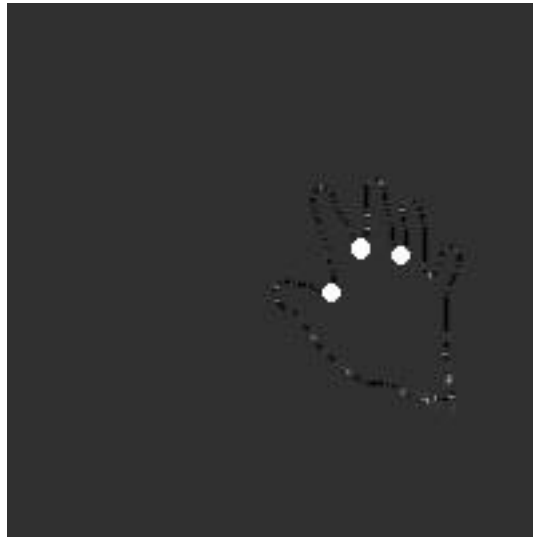


FIGURE 29 – Points d'intérêts de la main par l'opérateur de Harris.

Pour chaque image consécutive :

5.1.3 Seuillage sur la boîte 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.

Exemple de deux histogrammes selon une luminosité et un contraste différent : (Figure 30).

Comme l'atteste le tableau ci-dessus (Figure 30), nous avons affaire à des histogrammes bimodaux, même si c'est moins prononcé dans le second exemple

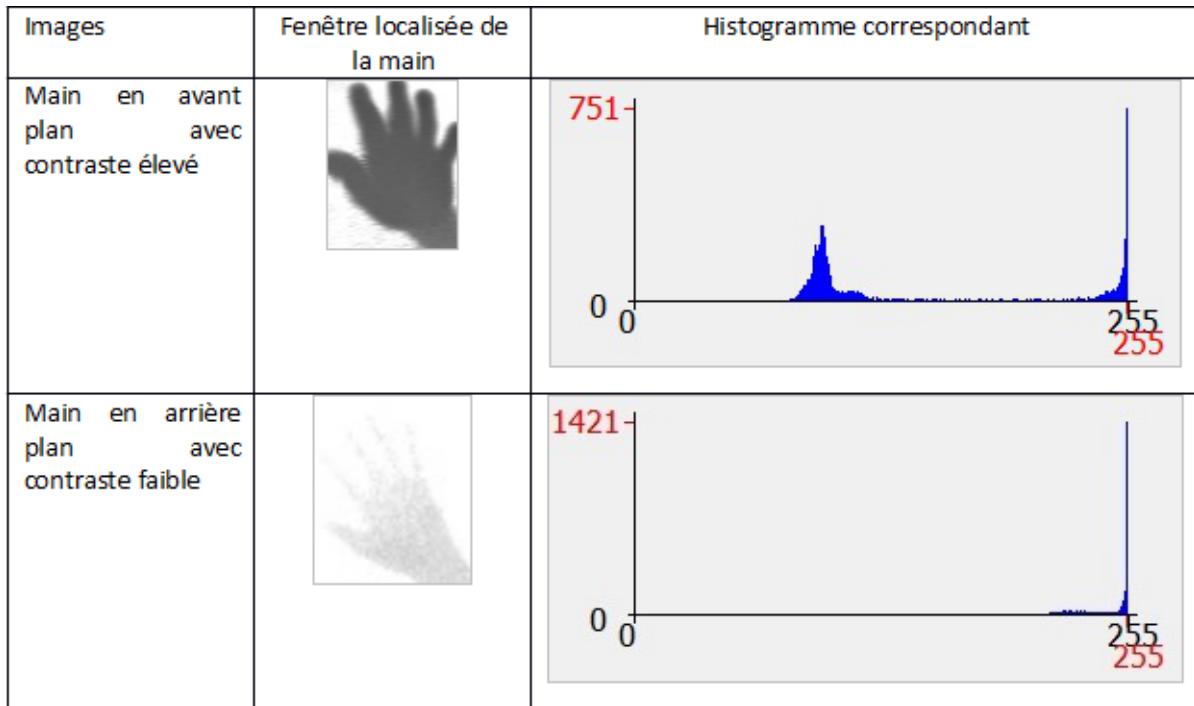


FIGURE 30 – Histogrammes selon la luminosité et le contraste.

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 on 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.

$$Thresh = mode + n$$

Ca 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.

Conclusion

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