Florent NOSARI, Oussama ZEKRI

22-mai-17

|  |
| --- |
| Approximation polygonale et comparaison des formes  Initiation à la recherche |



Equipe : ADAGIo, Loria

Encadrant : Mme Hoai Diem Phuc Ngo

|  |
| --- |
| Table des matières  [1. Remerciements 4](#_Toc483172621)  [2. Introduction 5](#_Toc483172622)  [3. Simplification polygonale 6](#_Toc483172623)  [3.1. Algorithme KiMPA 6](#_Toc483172624)  [3.2. Algorithme de Douglas-Peucker 8](#_Toc483172625)  [3.3. Détection des points dominants 10](#_Toc483172626)  [3.4. Implémentation de l’algorithme KiMPA 12](#_Toc483172627)  [3.5. Amélioration avec l’algorithme Douglas-Peucker 13](#_Toc483172628)  [4. Représentation et comparaison des formes 15](#_Toc483172629)  [4.1. Représentation 15](#_Toc483172630)  [4.2. Comparaison 16](#_Toc483172631)  [4.3. Résultats 19](#_Toc483172632)  [5. Conclusion 23](#_Toc483172633)  [6. Références 25](#_Toc483172634)  [7. Annexes 26](#_Toc483172635)  [7.1. D’autres méthodes de comparaisons 26](#_Toc483172636)  [7.2. Turning function : algorithmes de calcule des angles 28](#_Toc483172637) |
|  |

# Remerciements

Nous tenons à remercier toutes les personnes qui ont contribué au succès de notre travail de recherches et qui nous ont aidé lors de la rédaction de ce rapport.

Tout d'abord, nous adressons nos remerciements à notre responsable, Mme Hoai Diem Phuc Ngo, Maitre de conférences à la Faculté des Sciences de Vandœuvre-lès-Nancy et membre de l’équipe ADAGIo au Loria, qui nous a beaucoup aidé et conseillé durant nos recherches. Par la même occasion nous remercions l’équipe ADAGIo qui nous a fourni les fichiers et programmes nécessaires à nos recherches.

Nous remercions également tous les responsables du master informatique qui nous ont permis de découvrir le monde de la recherche à travers ce module d’initiation à la recherche.

Enfin, nous tenons à remercier toutes les personnes qui nous ont conseillé et relu lors de la rédaction de ce rapport : nos camarades de promotion, M. Nicolas Ray (enseignant-chercheur au Loria) et nos amis.

# Introduction

Dans la vision par ordinateur il est très important de pouvoir identifier des formes, on peut citer l’exemple de la traduction du langage des signes de laquelle découle l’objectif de cette étude. Il existe cependant différentes manières de représenter les formes afin de pouvoir les comparer. Nous allons ici nous intéresser à la méthode de l’espace tangentiel (*turning function*) qui est relativement efficace et qui pourrait permettre d’avoir une reconnaissance quasi-instantanée. La motivation de cette étude repose sur la recherche d’un moyen de traduction en temps réel du langage des signes à partir de captures de ceux-ci.

Nous avons à notre disposition un ensemble d’images représentant des signes qui constituent notre base de connaissances ainsi qu’un moyen de créer leurs représentations sous forme de polygones (représentés par un ensemble de points) à l’aide de divers programmes tiers.

Dans la première partie nous allons décrire différentes méthodes de simplification de polygones afin de pouvoir les comparer plus rapidement et minimisant le bruit sur les formes à comparer.

Dans un second temps nous allons détailler la méthode de comparaison tout en proposant une implémentation.

Dans la première partie de cette étude nous allons définir les caractéristiques de la fonction de comparaison de formes qu’il nous faut. Puis dans un second temps nous allons étudier différentes représentations possibles en mettant en évidence leurs caractéristiques. Nous donnerons quelques résultats (efficacité et performances) de notre implémentation en mettant en concurrence plusieurs méthodes de simplifications et en la comparant aux résultats que l’on obtiendrait avec une méthode d’apprentissage.

Pour conclure nous ferons le point sur nos résultats et expliquerons ce que nous avons appris lors de cette initiation à la recherche mais aussi ce que cela nous a apporté.

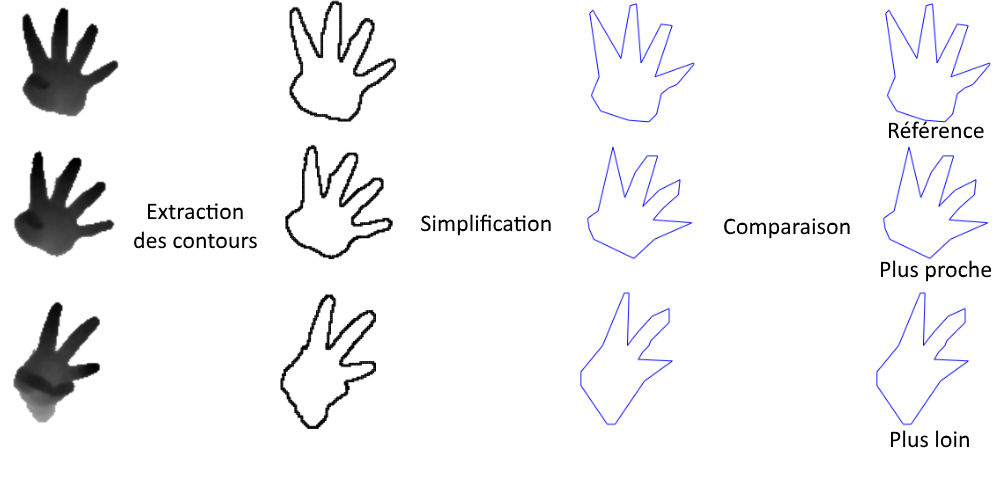


Figure : Processus de comparaison de formes

# Simplification polygonale

Lorsque l’on représente une image sous forme de polygone, il est très probable que celui-ci soit très complexe à analyser, en effet il se peut que le traitement coûte très cher en temps de d’exécution et en mémoire. Pour résoudre ce problème il est nécessaire de simplifier la forme afin d’effectuer le traitement sur une forme plus simple et avec le moins de bruit possible. La difficulté de cette simplification réside dans le fait qu’il faut tout de même garder les mêmes caractéristiques importantes que la forme originale.

Dans cette partie nous allons décrire deux méthodes de simplification polygonale, l’une est basée sur la cinématique et l’autre sur la géométrie, leur point commun est qu’elles gardent toutes deux la même représentation de polygone en sortie (polygone sous formes de liste de points). A l’issue de cette partie nous aurons mis au point un algorithme qui nous servira à effectuer une comparaison plus rapide qu’avec la forme de base.

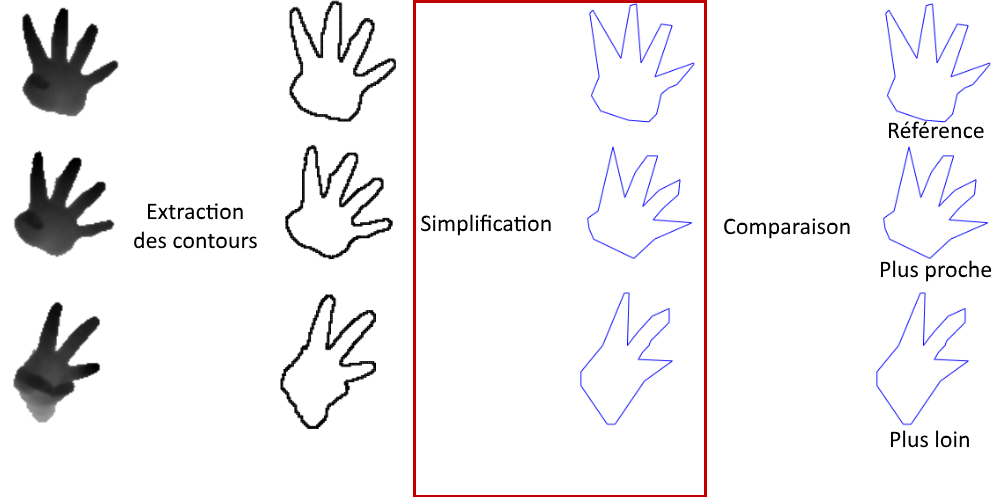


Figure : Etape de simplification dans le processus de comparaison

## Algorithme KiMPA

KiMPA [1] est une méthode de simplification polygonale qui associe à chaque sommet d’un polygone un niveau d’importance. La valeur du niveau d’importance est calculée par une méthode cinématique.

On caractérise un point de l’ensemble des sommets d’un polygone par :

* Sa vélocité : la vélocité d'un sommet est la vitesse de variation de la distance par angle :
* son accélération  l'accélération d'un sommet, est la vitesse de changement de vélocité par angle

Avec :

* est la distance entre le sommet v et le barycentre du polygone
* est la valeur de l’angle de v (Figure 3)

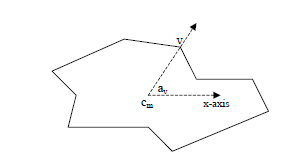


Figure Calcule de l'angle [1]

#### Exemple :

Pour le polygone de la Figure 4 Exemple [1] il serait préférable d'inclure les points et dans l’ensemble du résultat plutôt que d'inclure et, si l'un de ces trois sommets doit être éliminé.

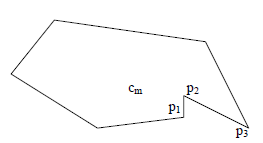


Figure Exemple [1]

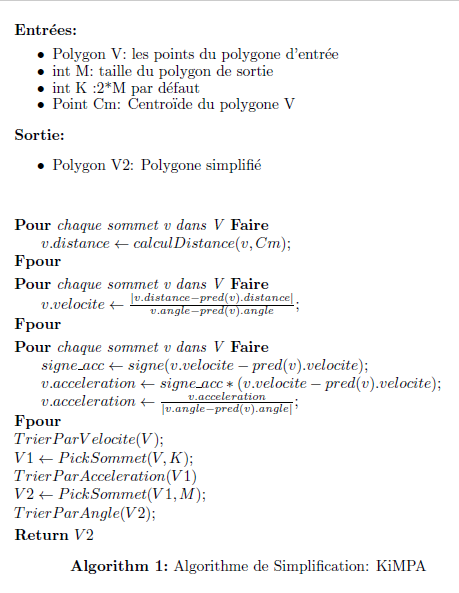
|  |  |  |
| --- | --- | --- |
| **Point** | **Vélocité** | **Ordre d’importance** |
| **P1** | 0.0013 | 3 |
| **P2** | 0.0068 | 2 |
| **P3** | 0.711 | 1 |

Plus la vélocité d'un sommet est importante, plus le polygone est pointu à ce sommet.

#### Principe de l’algorithme :

Afin de différencier les sommets, tous doivent être triés en fonction de leurs vélocités et accélérations. La première étape consiste à sélectionner les top-K des sommets dans l'ordre de vélocité descendant. Si le nombre K n'est pas spécifié par l'utilisateur, la valeur par défaut 2 × | PA | peut être utilisé. Parmi les K sommets, le top- | PA | sommets sont sélectionnés comme les sommets du polygone simplifié | PA |.

En traitant les sommets de cette manière, les sommets les plus « accélérés » (les plus importants) apparaissent dans le polygone approché puisqu'ils sont les sommets où le polygone d'origine a ses caractéristiques de forme.



#### Complexité :

L’algorithme de simplification KiMPA a une complexité de pour n’importe quel ensemble des n sommets de V qui est engendrée par l’opération de tri de V par vélocité.

## Algorithme de Douglas-Peucker

L'algorithme de simplification de ligne classique de Douglas-Peucker [2] est reconnu comme étant celui qui fournit les meilleures représentations perceptuelles des lignes d'origine.

Il peut cependant produire un polygone simplifié qui n'est pas topologiquement équivalent à l'original constitué d’une partie de ses sommets.

#### Principe :

L’algorithme de Ramer-Douglas-Peucker sert à simplifier un polygone ou une polyligne par la suppression de nœud. Il est beaucoup utilisé en compression de données vectorielles et en généralisation cartographique.

Une polyligne (n nœuds) est simplifiable et remplacée par une ligne simple (deux nœuds) si la distance de son nœud le plus éloigné de la droite formée par les extrémités de la polyligne est inférieure à un seuil.

#### Algorithme :

Au début on doit sélectionner les bornes A, B (deux sommets) du polygone et on choisit une distance de seuil.

A chaque étape, on parcourt les nœuds entre A et B et on calcule Distance (P[i],[AB]).

On choisit le sommet le plus éloigné.

S’il n’y a pas de nœuds entre A et B l’algorithme se termine.

Si cette distance est inférieure au seuil, on supprime les points entre les A et B.

Si cette distance est supérieure au seuil, on appelle récursivement l’algorithme sur les deux sous parties : ([A, Point distant] ; [Point distant, B].

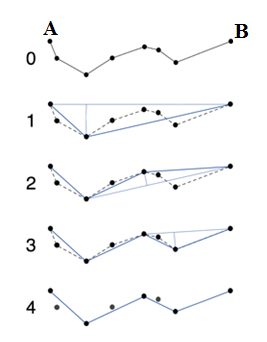


Figure Exemple de simplification par la méthode [8]

|  |  |
| --- | --- |
| **Distance** | **Valeur** |
| **AB** | 4.5 |
| **P1,AB** | 0.4 |
| **P2,AB** | **1** |
| **P3,AB** | 0.6 |
| **P4,AB** | 0.2 |
| **P5,AB** | 0.3 |
| **P6,AB** | 0.7 |

Le point P2 est le plus éloigné du segment AB donc on applique l’algorithme de Douglas Peucker sur les deux parties [A, P2] et [P2, B].

#### Complexité :

  L'algorithme se finit forcément puisque dans le pire des cas il n’y aura pas de simplification et chaque branche formée par la récursion s'achèvera lorsque les bornes ne seront pas séparées par un nœud

La complexité est en O (n ln(n)) puisqu'à chaque itération le nombre de branches de récurions est multiplié par deux et tous les nœuds sont visités.

## Détection des points dominants

Les points dominants [3] d’un polygone sont définis comme les points avec la courbure locale maximum. Ces points contiennent des informations qui sont suffisant pour caractériser la courbure d’une ligne discrète.

La détection des points dominants est utilisée pour simplifier des polygones ainsi que les domaines de vision par ordinateur.

La méthode classique de détection des points dominants a révélé des problèmes au niveau du temps de calcul, des nombres de paramètres, du choix du point de départ et les mauvais résultats sur les contours bruités.

#### Principe et algorithme :

La décomposition d’un contour C en Maximal Blurred Segments donne des informations importantes concernant sa structure géométrique. La largeur de ces segments permet de travailler sur plusieurs échelles et de considérer les contours avec bruits.

Pour cela, l’algorithme prend en entrée la largeur v puis calcule MBSv(C). Les points dominants de C sont localisés dans les zones communes des maximal blurred segments.

Une séquence de maximal blurred segments de largeur 1.4 d’un contour avec des pixels gris (voir figure 4).

Chaque rectangle rouge entoure les pixels d’un maximal blurred segment de largeur 1.4.

Les points avec un cadre bleu se situent dans les zones communes de ces segments.

|  |
| --- |
| Figure Maximal Blurred Segments [3] |

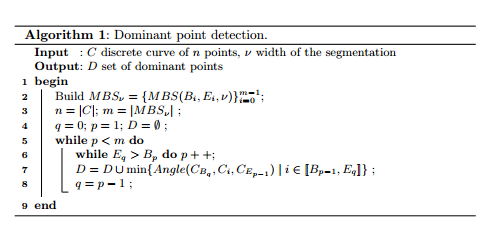
La méthode consiste à trouver la plus petite zone commune induite par des maximal blurred segments successives de largeur v qui ont des pentes croissantes ou décroissantes.

Cette zone contient des candidats de point dominant puisqu’ils ont le plus petit angle ROS.

Par une stratégie heuristique le point dominant est identifié comme le point de milieu de cette zone.

Cette heuristique est efficace mais ne conduit pas toujours à une solution optimale pour le problème de simplification polygonale.

Une amélioration de cette méthode est présentée dans l’algorithme [4] suivant et qui donne un résultat beaucoup plus optimal :



Cet algorithme propose une nouvelle stratégie de prise de décision de point dominant dans les zones communes. C’est la mesure d’angle entre le point considéré et les deux extrémités à gauche et à droites des maximal blurred segments qui forment cette zone d’intersection par ce point.

Plus l’angle est petit, plus la courbure est importante donc plus le point est dominant.

En conséquence, le point dominant est identifié comme le point, dans la zone commune, ayant une mesure minimale locale d'angle.

#### Complexité :

La complexité de cette méthode est , elle dépend essentiellement de la complexité de décomposer un contour de n points en maximal blurred segments.

## Implémentation de l’algorithme KiMPA

L’étape de simplification est une phase importante dans le processus de l’approximation polygonale et la comparaison de forme. Elle permet ainsi de réduire le nombre de points d’un contour afin d’assurer un calcul rapide et simplifié.

Dans notre projet nous avons choisi d’utiliser la méthode KiMPA qui nous a fourni une représentation réduite d’une forme principale.

Les polygones sont simplifiés selon les valeurs de vélocité et d’accélération des sommets avec le respect du barycentre du polygone.

La vélocité et l’accélération sont utilisées pour exprimer le niveau d’importance d’un sommet et dans le résultat de l’algorithme on ne conserve que ceux qui ont un niveau d’importance élevé.

#### Avantage :

* Algorithme simple à implanter
* Garder les informations cruciales sur un polygone simplifié

#### Inconvénient :

* Difficulté de traitement sur les polygones avec un grand nombre de sommets
* Pratiquement, sur un ensemble de points successifs très proches il y aura un problème pour détecter le sommet important

#### Résultats :

Voici des résultats de l’algorithme KiMPA avec variation du nombre de points du polygone de sortie.

#### 

Figure  : Exemple 1

La figure 5 montre la comparaison du résultat (contour en vert) de l’algorithme KiMPA sur le polygone d’entrée (en rouge) qui contient 411 points avec une variation du nombre de points du polygone de sortie.

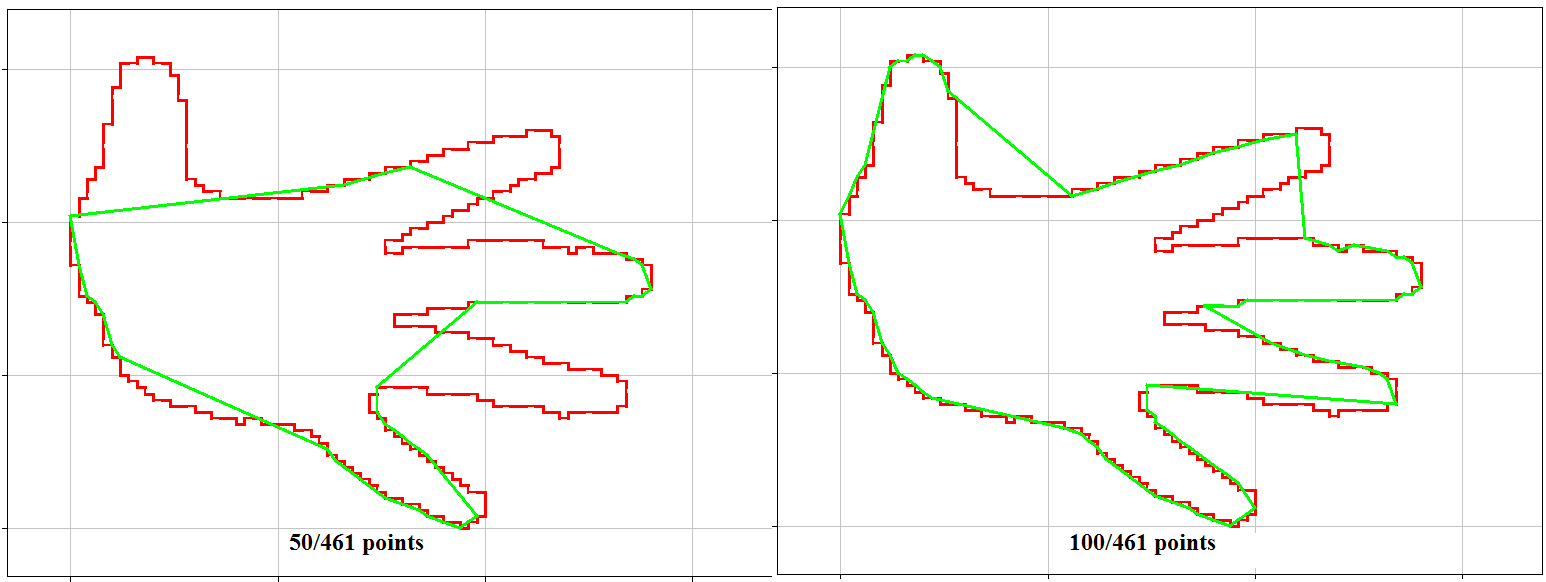


Figure : Exemple 2

De même, la figure 5 compare une simplification d’un polygone de 461 points mais en variant le nombre de sommets du polygone résultat.

#### Observations :

D’après les résultats précédents, on remarque que l’algorithme ne détecte pas toujours les informations importantes d’un contour (points significatifs), cela est du à des valeurs étranges de vélocité et d’accélération pour des points très proches (distance entre 2 points égal à 1).

Donc pour obtenir un meilleur résultat on doit augmenter le nombre de points du polygone de sortie.

## Amélioration avec l’algorithme Douglas-Peucker

Afin d’améliorer notre algorithme, nous proposons de faire un prétraitement sur le polygone d’entrée pour détecter les points successifs ont des coordonnées très proches.

Pour cela, nous avons utilisé l’algorithme de Douglas-Peucker dont le principe est déjà énoncé.

Nous obtenons par la suite les résultats suivants :

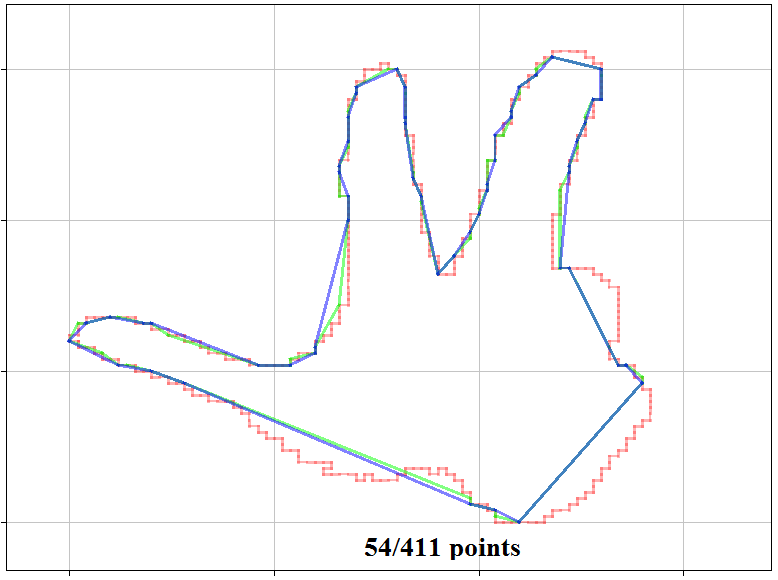


Figure : Résultat exemple1

Dans la figure 7 on montre le même exemple de la figure 5 mais amélioré avec l’algorithme de Douglas Peucker (en bleu) et qui donne en résultat un polygone de 54 sommets.

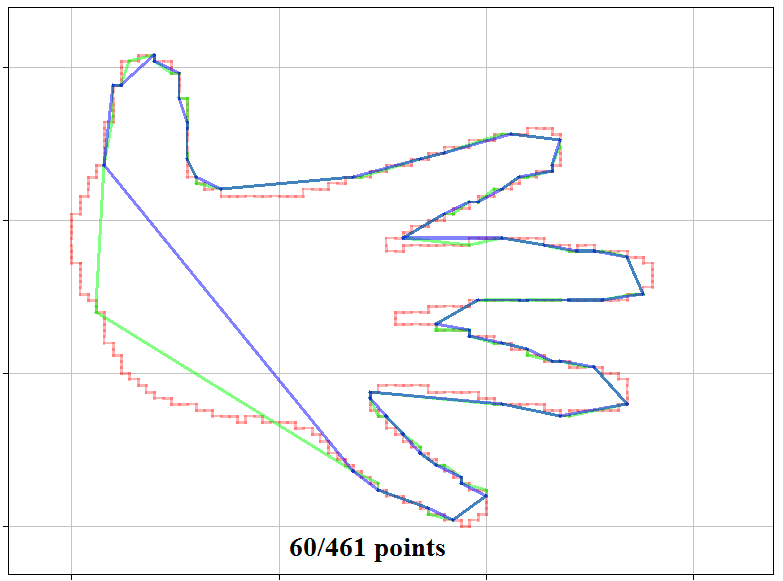
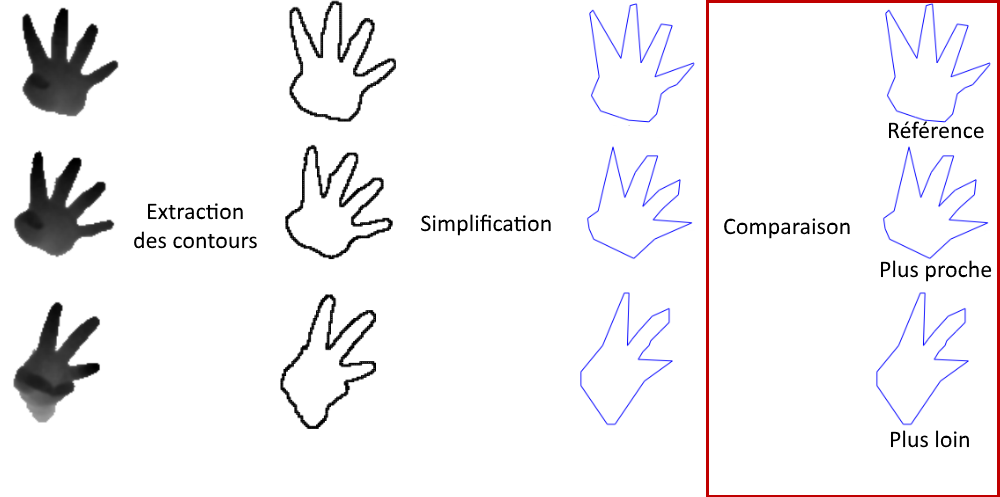


Figure : Résultat exemple 2

La figure 8 est le résultat de la méthode de simplification de Douglas Peucker sur l’exemple de la figure 6, en résultat nous obtenons un polygone de 60 points.

# Représentation et comparaison des formes

Dans cette seconde partie nous allons étudier la méthode de l’espace tangentiel (*turning function*) décrite dans [5]. Cette méthode se fait en deux étapes, la représentation puis la comparaison. Nous avons sélectionné cette méthode parmi les classes de méthodes étudiées (recherches disponibles en annexe) pour ses propriétés. Nous expliquerons dans un premier temps l’étape de représentation en détaillant les propriétés qui nous intéressent puis nous expliquerons comment s’effectue la comparaison.



## Représentation

Le principe de cette méthode est de représenter le polygone sous la forme d’une courbe qui décrit la forme de celui-ci. L’axe des abscisses représente la taille des segments (taille exprimée en pourcentage du périmètre du polygone) et l’axe des ordonnées représente la variation des angles (angles orientés). La complexité de cet algorithme est .

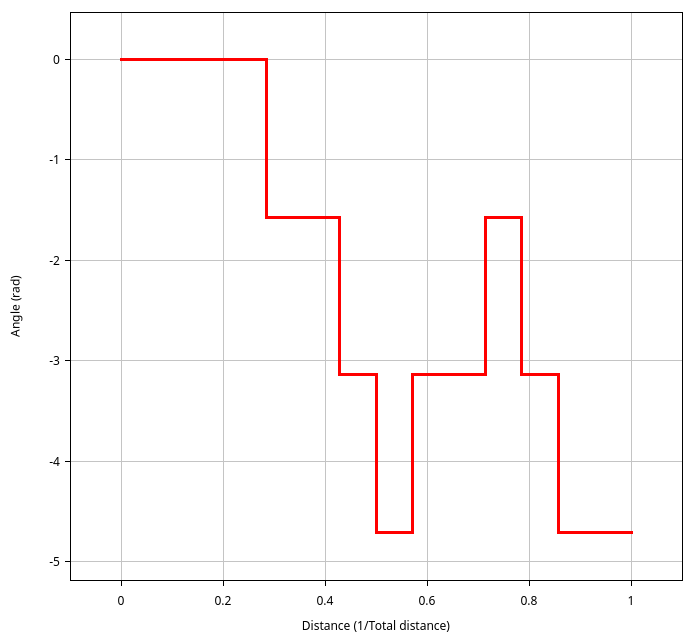
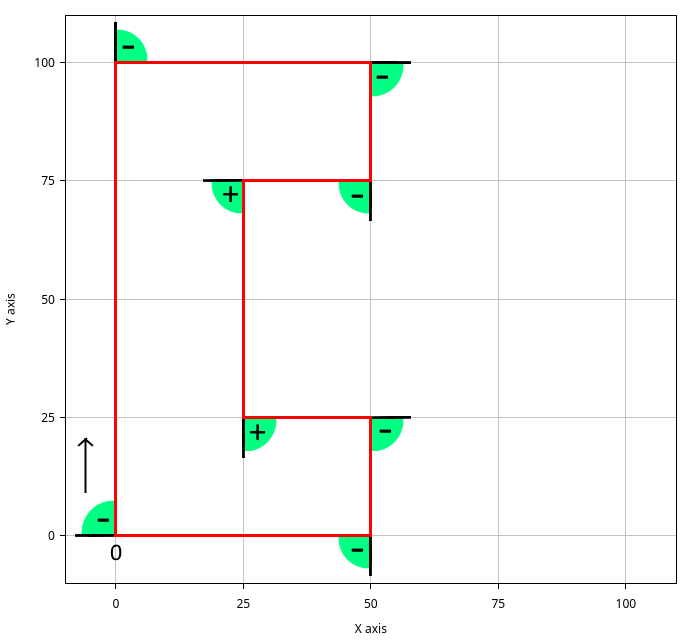
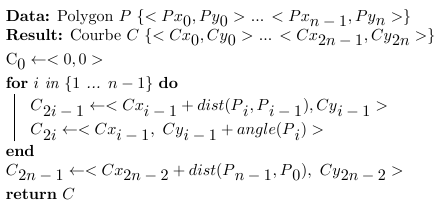


Figure 11 : Un polygone et sa représentation dans l’espace tangentiel

Voici ce à quoi peut ressembler le pseudo algorithme :



Algorithme 1 : Calcule de l'espace tangentiel

À ce stade on peut déjà constater plusieurs propriétés.

* L’invariance la translation : les coordonnées des points du polygone ne sont pas prises en compte pour créer cette représentation, seule la valeur les angles et la longueur des segments sont requis.
* L’invariance au changement d’échelle : la longueur des segments est exprimée en un ratio ce qui permet d’outrepasser la taille réelle du polygone.

## Comparaison

Pour comparer deux polygones avec cette approche, il faut calculer la différence entre les deux aires sous la courbe obtenue (Figure 13 : Comparaison de 2 formes à l’aide de la turning function. Il faut cependant prendre en compte la possibilité que le point de départ ait changé (Figure 12). Il faut donc pour cela calculer le minimum de la différence des aires en changeant le point de départ d’un des deux polygones à chaque fois.

Pour calculer la distance entre un polygone et un polygone il faut effectuer le calcul suivant :

avec et représentations respective des polygone et .

On a donc pour complexité globale  si on considère que la méthode de calcul d’aire est négligeable, sinon (dans le cas où l’on calcule l’aire formée par tous les rectangles pour calculer l’aire globale).

|  |  |
| --- | --- |
| 0° | 90° |
|  |  |
|  |  |

Figure 12 : Un polygone et sa représentation dans l’espace tangentiel en fonction de l’angle

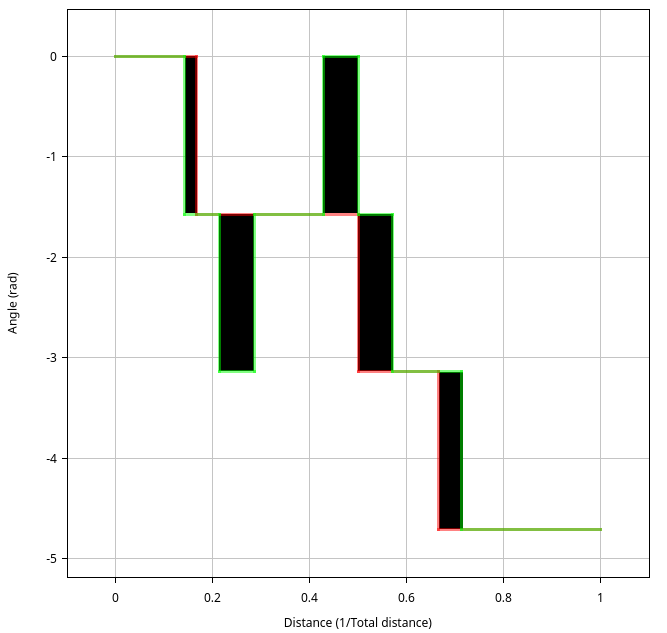
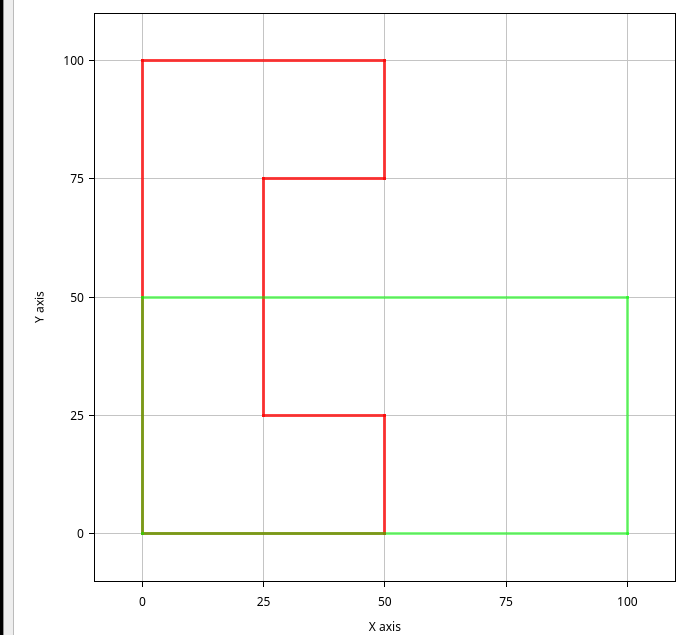


Figure 13 : Comparaison de 2 formes à l’aide de la turning function

Avec cette façon de comparer deux courbes on apporte les propriétés suivantes :

* La méthode de comparaison devient une métrique

La méthode satisfait donc les conditions suivantes :

Soit la fonction de comparaison et 3 polygones.

* + La distance ne peut être négative
  + La distance entre 2 éléments identiques est nulle et réciproquement (identité des indiscernables)
  + La distance est symétrique (ne dépend pas de l’ordre)
  + La distance respecte le principe de l’inégalité triangulaire
* Invariance au choix point de départ : le fait de calculer le minimum de l’aire entre les courbes permet de ne jamais dépendre du point de départ dans le polygone. On peut en déduire également la propriété suivante.
* Invariance à la rotation : le fait que les coordonnées des points ne changent en rien la courbe, on peut donc dire qu’une rotation ne fait que changer le point de départ.

Avec l’étude de différentes méthodes, nous avons pu déterminer la meilleure solution pour notre problème. En effet nous avons constaté que la méthode de l’espace tangentiel possédait des caractéristiques très intéressantes pour notre problème de comparaison et permettait d’avoir des résultats relativement précis.

## Résultats

Après avoir implémenté les méthodes choisies nous avons procédé à des tests pour évaluer notre solution. L’implémentation a été faite à l’aide de la bibliothèque DGtal (http://dgtal.org/) développée par divers laboratoires de recherches français. Nous avons utilisé en particulier deux programmes pour la récupération des points à partir d’une image PNG.

Les tests ont été réalisés sur un ordinateur équipé d’un processeur Intel® Core I5-66000K, doté de 8Go de RAM et du système d’exploitation Ubuntu basé sur le noyaux GNU/Linux.

Les images utilisées ont été fournies par l’équipe ADAGIo et sont issues de captures effectuées à l’aide d’une caméra Kinect, elles représentent toutes des signes du langage des signes.

Le code source de nos programmes est disponible via git à l’adresse suivante : https://gitlab.com/nosariflorent/polygon.

### Notre implémentation



Figure 14 : Calcule des distances entre différents signes.

Sur ces résultats on peut constater plusieurs choses. Tout d’abord on voit que la distance entre deux formes égales est nulle, ceci respecte donc la première propriété d’une métrique. Ensuite on constate que la distance entre deux formes est toujours la même peu importe l’ordre, ce qui respecte donc la troisième propriété d’une métrique.

En ce qui concerne les données en elles-mêmes on peut voir que la forme la plus proche est toujours la forme avec plus ou moins un doigt en plus.



Figure 15 : Calcule des distances entre différentes représentations d’un même signe ( la couleur grise représente le 50ème centile).

Sur ces résultats on peut constater que bien que la méthode de comparaison soit efficace, on doit tout de même posséder une base. En effet on constate que la distance au 50ème centile est tout de même aux alentours de 0.7, soit donc une distance assez importante à la vue des résultats précédents sur des signes différents.



Figure 16 : Calcule des distances entre différentes représentations d'un même signe ayant subi une rotation lors de la capture

On constate ici que les distances sont plus basses mais il en reste des très élevés (distance similaire à la distance entre deux signes différents). Le problème est que cela peut venir de plusieurs choses. En effet il se peut qu’il y ait des erreurs lors de l’extraction des contours et/ou lors de la simplification.



Figure : Calcule des 10 plus proches cumulés pour chaque classe en excluant les comparés avec 500 fichiers.

Nous pouvons voir avec ces résultats que la majorité des formes sont bien identifiées et sont donc bien affecté à la bonne classe. Mais on constate que les résultats diminuent selon la complexité du signe.

En ce qui concerne la rapidité nous avons constaté que le temps moyen avoisine les 45 secondes pour 500 comparaisons à raison d’une moyenne de 37 points par polygone soit environ 100 millisecondes par comparaison. Ce temps passe à 99 secondes lorsque la moyenne des points passe à 50, soit 198 millisecondes puis lorsque la moyenne des points passe à 75 points nous avons un temps très élevé de 340 secondes, soit 680 millisecondes. Ce temps semble plutôt élevé ce qui soulève donc le problème de l’optimisation lors de l’implémentation. Mais avec ces tests nous voyons bien l’importance de l’étape de simplification qui réduit drastiquement le temps de traitement.

### Changement de méthode de simplification

Afin d’avoir une autre mesure lors de nos résultats, l’équipe ADAGIo nous a fourni un programme de simplification qui utilise la méthode de détection des points dominants détaillé en 1.3.3 ci-dessus. Ce programme nous a permis de mettre en évidence la différence des résultats en fonction de la méthode de simplification choisie mais aussi de sa qualité.

### Apprentissage

Pour comparer la fiabilité et la rapidité de la méthode de comparaison que nous avons implémenté à d’autre méthodes, nous avons choisi de tester la méthode par apprentissage décrite en annexe 1.7.1. Pour cela nous avons implémenté les descripteurs suivant :

Soit le polygone et l’enveloppe convexe de

* mesure de la compacité de
* mesure de la compacité de
* mesure de la convexité de
* Moyenne des angles de
* Variance des angles de
* Distance minimum entre un point de et le barycentre de
* Distance maximum entre un point de et le barycentre de
* Différence entre la distance minimum et maximum entre un point de et le barycentre de
* Variance de la taille des segments de

En ce qui concerne l’algorithme en lui-même nous avons choisi d’utiliser l’algorithme des k-moyennes [6] (*k-means*) qui consiste en la création de *clusters* à partir de données que l’on donne en entrée.

Malheureusement les tests ont été peut concluants (taux classification correct inférieur à 30%) et n’ont pas permis de comparer les performances.

# Conclusion

Pour conclure sur nos résultats on peut voir que la méthode de simplification implémentée permet de garder beaucoup d’informations sur le polygone. On a vu cependant qu’il fallait effectuer un traitement préalable pour supprimer le plus possible de points successifs trop proches les uns les autres et sur la même droite afin d’avoir des résultats optimaux pour le traitement des images de la base fournie. D’autre part, on peut voir que la méthode de comparaison que nous avons implémentée est plutôt efficace et donne des résultats satisfaisants mais nécessite d’être optimisée lors de l’implémentation afin de gagner en rapidité. Il faut cependant noter qu’elle dépend fortement de la façon dont le polygone de base a été simplifié.

En ce qui concerne le travail de recherche, ce module nous a permis de découvrir dans le détail le processus de simplification et comparaison des formes qui est très utilisé dans le domaine de traitement d’image et de vision par ordinateur. A travers ce projet nous avons passé par les différentes étapes d’un travail de recherches qui commence par la compréhension du sujet et une étude d’articles, puis une recherche des solutions possibles des différents problèmes liés à notre sujet. Concernant la partie pratique de notre travail, nous avons implémenté différente applications afin de visualiser chaque étape de nos recherches.

# Références

|  |  |
| --- | --- |
| [1] | G. G. U. G. a. ¨. U. Ediz S¸aykol, *KiMPA: A Kinematics-Based Method for.* |
| [2] | «Algorithme de Douglas-Peucker,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Algorithme\_de\_Douglas-Peucker. |
| [3] | N. H. D.-R. I. Ngo P., «Eﬃcient dominant point detection based on discrete curve structure,» *IWCIA ’15. LNCS, vol. 9448,* pp. 143-156, 2015. |
| [4] | H. N. I. D.-R. Phuc Ngo, «Efficient dominant point detection based on discrete». |
| [5] | L. P. C. Esther M. Arkin et K. K. a. J. S. B. M. Daniel P. Huttenlocher, «An Efficiently Computable Metric for Comparing Polygonal Shape,» *IEE Transaction on pattern analysis and machine intelligence, vol 13, N0. 3,* Mars 1991. |
| [6] | M. N. M. A. K. J. a. P. J. Flynn, «Data clustering : A review,» *ACM Computing Surveys vol. 31,* pp. 264-323, 3 November 1999. |
| [7] | M. R. Teague, «Image analysis via the general theory of moments,» *J. Opt. Soc. Am., vol. 70, no. 8, pp. 920–930,* 1980. |
| [8] | T. F. B. C. a. D. D. Robert Osada, «Shape Distributions,» *ACM Transactions on Graphics, Vol. 21, No. 4,* p. 807–832, Octobre 2002. |
| [9] | H. Wiederhold P. and Reyes, «Relative Convex Hull Determination from Convex Hulls in the Plane,» *IWCIA ’15. LNCS, vol. 9448,* pp. 46-60, 2015. |
| [10] | S. T. Thai V. Hoang, «A geometric invariant shape descriptor based on the,» MICA, UMI 2954, Hanoi University of Technology, Hanoi, Vietnam 2LORIA, UMR 7503, Universite Nancy 2, 54506 Vandœuvre-lès-Nancy, France, 13 Septembre 2010. [En ligne]. |
| [11] | M. Zhang, «COS598b Geometric Modeling,» 2000. [En ligne]. Available: http://www.cs.princeton.edu/courses/archive/spr00/cs598b/lectures/polygonsimilarity/polygonsimilarity.pdf. [Accès le Mars 2017]. |
| [12] | J.-O. L. Bertrand Kerautret, «Meaningful Scales Detection: an Unsupervised Noise,» 07 Mai 2014. [En ligne]. Available: http://www.ipol.im/pub/art/2014/75/. |

# Annexes

## D’autres méthodes de comparaisons

### Descripteurs

Les méthodes de classification consistent à représenter un polygone par des descripteurs. Ces méthodes permettent de les classer avec différentes techniques de classification utilisant l’apprentissage avec comme vecteur d’entrée la liste de ses descripteurs. Une fois l’apprentissage fini est vérifié il suffit de donner un nouveau polygone à l’algorithme pour qu’il détermine à quelle classe appartient celui-ci en se basant sur ses descripteurs.

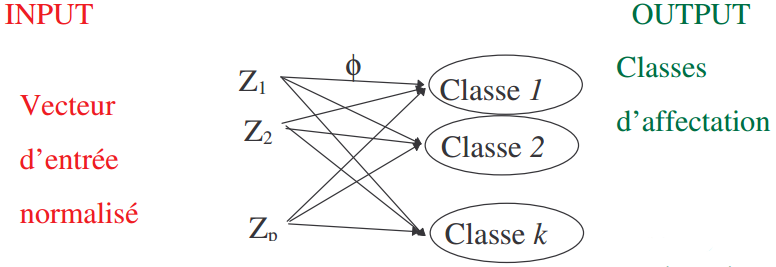


Figure 18 Représentation du fonctionnement de l'algorithme

Chaque descripteur est un nombre ou un vecteur de nombre représentant une propriété du polygone, on peut citer quelques exemples de ces descripteurs :

* Le nombre de points
* L’aire
* Le périmètre
* Les angles
* La compacité
* Le barycentre
* La longueur des segments
* La convexité
* Les moments [7]
* Les signatures [8]
* L’enveloppe convexe [9]

Il y a aussi des descripteurs dans un espace continu, on peut citer par exemple :

* La transformé de Fourier [10]
* La transformé de Radon [10]
* La transformé de Mellin [10]

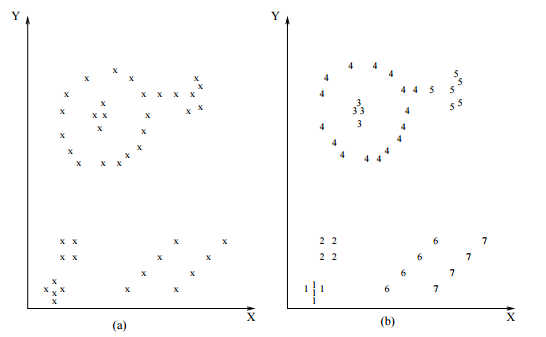


Figure  : Exemple avec l’algorithme k-means [6] avec des vecteurs de dimension 2 (source : [6])

Malgré l’efficacité de cette méthode, son principal problème est qu’elle nécessite l’apprentissage d’une base existante qui doit être importante et que le temps de traitement est relativement long.

### Graphes

La classe de méthodes utilisant les graphes consiste à représenter un polygone par un graphe dans lequel les sommets représentent les régions issues d’une décomposition du polygone, et la description du polygone est représentée par les propriétés du graphe.

Il y a deux approches générales pour la décomposition :

* L’amincissement des régions pour aboutir au squelette du polygone
* La décomposition en régions représentées par des sommets avec le rapport de voisinage des régions représenté par les arcs.

Pour illustrer ce principe, nous allons détailler la deuxième approche à partir des explications données dans [11]. Dans un premier temps il faut diviser le polygone par des lignes parallèles à l’axe des abscisses de sorte à créer des polygones appelés morceaux tels que tout point d’un segment de 2 points du morceau parallèle à l’axe des abscisses fasse partie du morceau. Ensuite on construit un graphe orienté tel que :

* Chaque morceau constitue un sommet du graphe
* Si 2 morceaux ont plus d’un point en commun alors on les relie
* Un sommet est parent d’un autre si le morceau correspondant est plus bas que le morceau correspondant à l’autre sommet.

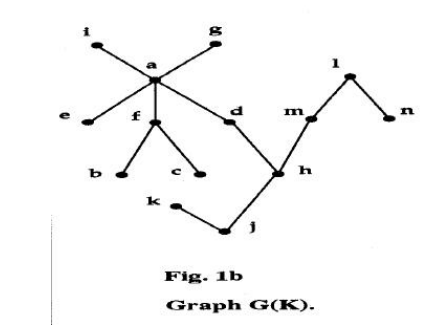
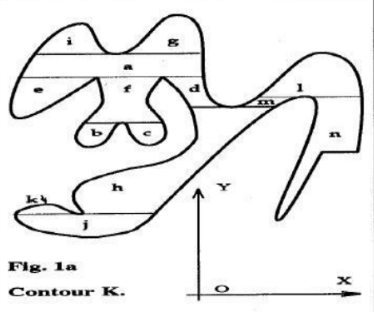
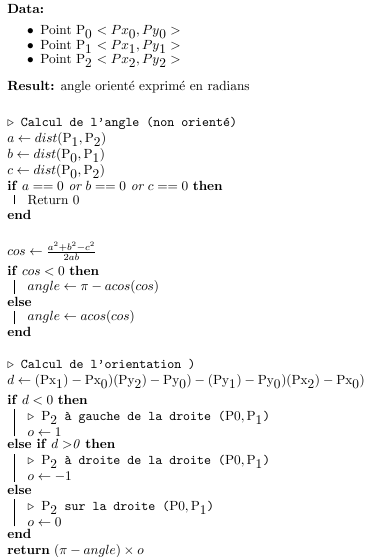


Figure 20 Représentation d’une forme par un graphe [11]

## Turning function : algorithmes de calcule des angles



Algorithme 2 : calcul de l'angle