N°d'ordre NNT : 2015LYSEio75

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA de Lyon

Ecole Doctorale N° 512
École Doctorale InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 11/07/2016, par :
Aurélie LEBORGNE

Appariement de formes basé sur une squelettisation hiérarchique

---

Devant le jury composé de :

LACHAUD, Jacques-Olivier, PR, Université de Savoie-Mont-Blanc Rapporteur
LOHOU, Christophe, PR, Université d'Auvergne Rapporteur
MONTANVERT, Annick, PR, Université Pierre-Mendès-France Examinatrice
NORMAND, Nicolas, MCF HDR, Ecole Polytechnique de l'Université de Nantes Examinateur

BASKURT, Atilla, PR, INSA-LYON Directeur de thèse
TOUGNE, Laure, PR, Université Lyon 2 Co-directrice de thèse
MILLE, Julien, MCF, INSA-Centre Val de Loire Co-encadrant

Aux personnes qui me sont chères...

Remerciements

Un grand merci à vous tous qui m'avez accompagnée tout au long de cette thèse.

TABLE DES MATIÈRES

TABLE DES MATIÈRES **vii**

LISTE DES FIGURES **xi**

INTRODUCTION GÉNÉRALE **1**

1 NOTIONS ET ALGORITHMES DE BASE DE GÉOMÉTRIE DISCRÈTE **5**
- 1.1 IMAGE NUMÉRIQUE EN DEUX DIMENSIONS . . . 7
- 1.2 FORME DANS UNE IMAGE NUMÉRIQUE . . . 8
- 1.3 TOPOLOGIE ET POINT SIMPLE . . . 10
  - 1.3.1 Composante connexe . . . 11
  - 1.3.2 Point simple . . . 12
  - 1.3.3 Suivi de contours . . . 15
- 1.4 SEGMENT DISCRET ENTRE DEUX PIXELS . . . 18
- 1.5 DISTANCES DISCRÈTES ET TRANSFORMATIONS EN DISTANCE . . . 18
  - 1.5.1 Transformée en Distance Euclidienne au Carré (*SEDT*) . . . 21
  - 1.5.2 Transformée en Distance Euclidienne Séquentielle Signée . . . 25
- 1.6 BOULES MAXIMALES DISCRÈTES ET AXE MÉDIAN DISCRET . . . 26

CONCLUSION . . . 29

2 ÉTAT DE L'ART DE LA DESCRIPTION À L'APPARIEMENT DES FORMES PLANES **31**
- 2.1 SQUELETTES . . . 37
  - 2.1.1 Méthodes basées sur un amincissement topologique . . . 38
  - 2.1.2 Méthodes basées sur la carte de distance . . . 40
  - 2.1.3 Méthodes basées sur le diagramme de Voronoï . . . 42
- 2.2 HIÉRARCHISATION ET ÉLAGAGE DU SQUELETTE . . . 46
  - 2.2.1 Hiérarchisation par relation de parenté . . . 47
  - 2.2.2 Hiérarchisation par approche multirésolution . . . 51
  - 2.2.3 Hiérarchisation sous-entendue . . . 52
  - 2.2.4 Une conséquence : l'Élagage . . . 54
- 2.3 APPARIEMENT DU SQUELETTE . . . 55
  - 2.3.1 Graphe acyclique orienté (DAG pour *Directed Acyclic Graph*) . . . 56
  - 2.3.2 Graphe Relationnel Attribué (ARG) . . . 58
  - 2.3.3 Appariement par la construction d'un arbre itérativement . . . 60

2.3.4 Appariement tenant compte du contexte . . . 62
CONCLUSION . . . 62

3 SQUELETTISATION **65**
3.1 EXTRACTION DU SQUELETTE EUCLIDIEN DISCRET CONNECTÉ (DECS) . . . 69
3.1.1 Vue d'ensemble . . . 69
3.1.2 Axe Médian Discret Réduit (RDMA) . . . 70
3.1.3 Filtrage Laplacien de Gaussienne (LoG) . . . 70
3.1.4 Combinaison du RDMA et de la carte des crêtes . . . 73
3.1.5 Représentation par une adjacence de branches . . . 76
3.2 MÉTHODES UTILISÉES POUR COMPARAISON . . . 77
3.2.1 Méthode de Bertrand et Couprie (2014) : Amincissement parallèle basé sur les noyaux critiques . . . 78
3.2.2 Méthode de Choi et al. (2003) : Extraction du squelette Euclidien basé sur un critère de connexité . . . 78
3.2.3 Méthode de Siddiqi et al. (2002) : Squelette Hamilton-Jacobi . . . 79
3.3 RÉSULTATS ET COMPARAISONS . . . 81
3.3.1 Résistance au bruit . . . 81
3.3.2 Influence des paramètres . . . 84
3.3.3 Connexité . . . 84
3.3.4 Complexité . . . 84
3.3.5 Reconstruction . . . 88
3.3.6 Discussion . . . 90
3.4 EXEMPLES DE SQUELETTES . . . 93
CONCLUSION . . . 97

4 HIÉRARCHISATION DU SQUELETTE **99**
4.1 LISSAGE À AIRE CONSTANTE . . . 103
4.1.1 Équation d'évolution (Gage (1986),Dolcetta et al. (2002)) . . . 103
4.1.2 Discrétisation . . . 104
4.1.3 Ré-échantillonnage . . . 105
4.1.4 Obtention d'une forme discrète à partir du contour lissé . . . 106
4.1.5 Remplissage de la Forme Discrète . . . 106
4.2 MÉTHODES POUR LA CONSTRUCTION D'UN SQUELETTE HIÉRARCHIQUE . . . 106
4.3 MODÉLISATION DU SQUELETTE HIÉRARCHIQUE PAR UN HYPER-GRAPHE HIÉRARCHIQUE . . . 118
4.3.1 Définition . . . 118
4.4 RÉSULTATS . . . 119
4.4.1 Stabilité sous transformations affines . . . 119
4.4.2 Élagage du squelette . . . 121
4.4.3 Complexité . . . 123

CONCLUSION 123

5 APPARIEMENT ET RECONNAISSANCE DE FORMES **125**
5.1 MÉTHODOLOGIE 129
5.1.1 Mesure de dissimilarité entre deux hyper-sommets (méta-branches) provenant de deux hyper-graphes : procédure d'initialisation (1) 129
5.1.2 Appariement des hyper-sommets grâce à l'algorithme hongrois (Kuhn 1955) : procédure d'initialisation (2) 139
5.1.3 Algorithme et exemple de l'étape d'initialisation 140
5.1.4 Raffinement de l'appariement entre hyper-graphes : procédure de mise à jour . . 140
5.1.5 Mesure de dissimilarité globale entre deux hyper-graphes 143
5.2 RÉSULTATS 144
5.2.1 Méthodes utilisées lors de la comparaion 144
5.2.2 Complexité 146
5.2.3 Résultats expérimentaux 148
5.2.4 Application sur une base de données de feuilles d'arbres 155
5.2.5 Exemples d'appariements de feuilles d'arbres 157
CONCLUSION 159

CONCLUSION GÉNÉRALE **161**

BIBLIOGRAPHIE **167**

LISTE DES FIGURES

1 Représentation d'un peigne. 2
2 Schéma général d'un système de reconnaissance d'objets d'après Likforman-Sulem et Barney-Smith (2013). 2

1.1 Mise en évidence de la discrétisation lors de l'utilisation d'un appareil photo numérique. 7
1.2 Représentation d'un segment de droite (a) continu, (b) discret. 8
1.3 Illustration d'un pavage carré régulier permettant de représenter une image numérique. 8
1.4 Décomposition d'un pixel. Les pointels sont en bleu et les lignels en rouge. 9
1.5 (a) Voisinage 4-connexe, (b) Voisinage 8-connexe. 9
1.6 Exemples de chemins 4 et 8 connexes. 10
1.7 Illustration de la notion de forme 8-connexe. 10
1.8 Exemples de formes binaire. (a) clé, (b) chien et (c) papillon. 11
1.9 Illustration de la notion de composante 4-connexe et 8-connexe. 12
1.10 Illustration de la notion de point simple. 12
1.11 Indexation du voisinage d'un pixel. 13
1.12 Exemple d'un point $\boldsymbol{p}$ 8-simple mais non 4-simple. 14
1.13 Exemple de pointel de contour et de lignel de contour. 15
1.14 Motif de départ du suivi de contour. 16
1.15 Placement du quatromino sur l'image. 16
1.16 Illustration des différentes configurations du suivi de contour. 16
1.17 Décomposition de l'espace 2D en octants. 18
1.18 Représentation d'un segment discret dans l'octant 1. 20
1.19 Illustration de la SEDT obtenue à partir de Meijster et al. (2002) et de Coeurjolly et Montanvert (2007), (a) la forme, (b) la carte G résultant de la première étape, (c) la transformée en distance euclidienne au carré finale. 21
1.20 Représentation de la valeur associée à $\mathcal{A}_{y_2}^{x_4}(y_3)$. 23
1.21 Illustration de l'étape 2 concernant la quatrième colonne. Les points appartenant à l'enveloppe inférieure sont entourés en rouge. 25
1.22 Masques utilisés dans la 8SSED. 26
1.23 Illustration de la 8SSED obtenue à partir de Ye (1988), (a) initialisation, (b) étape 1 (déplacement des masques 1 et 2), (c) étape 2 (déplacement des masques 3 et 4). 28

1.24 (a) Illustration du concept de boule maximale (Coeurjolly et Montanvert (2007)), (b) Axe médian d'une feuille. 28

2.1 Illustration de formes variant considérablement en fonction des déformations (Yang et al. (2016)). 33
2.2 Le squelette est une représentation compacte intuitive du corps humain. 37
2.3 Représentation d'un serpent, associé à son squelette, suivant deux positions. . . . 37
2.4 Illustration du feu de prairie. 38
2.5 Exemple d'amincissement topologique parallèle. 40
2.6 Courbes de niveaux et représentation discrète associée 40
2.7 (a) Forme de reptile, (b) Représentation en trois dimensions de la carte de distance associée, (c) squelette obtenu à partir de la carte de distance. 41
2.8 Gradient autour (a) d'une crête, (b) d'un sommet. La distance est représentée en niveaux de gris. 42
2.9 Création d'un diagramme de Voronoï lors de la reproduction du Picarel. 43
2.10 Exemple de diagramme de Voronoï. 44
2.11 Exemple de squelette de Voronoï. Le bord de la forme est constitué par la ligne rouge, les points échantillonnés sont en bleu et le squelette en noir (Dardenne et al. (2009)). 44
2.12 Exemple de forme ayant une épaisseur paire (6 pixels) associée à son squelette (en jaune) centré mais épais (a), décentré mais fin (b). 45
2.13 (a) Squelette d'un rectangle, (b) Squelette du même rectangle auquel a été ajouté une perturbation sur le bord. 47
2.14 Illustration de la relation parent-enfant entre les branches pour le squelette de la forme encadrée en rouge. 48
2.15 Illustration de l'importance des branches pour Ogniewicz et Kübler (1995). . . . 49
2.16 (a) Choc du premier ordre, (b) Choc du deuxième ordre, (c) Choc du troisième ordre et (d) Choc du quatrième ordre. Les chocs sont représentés en rouge. 49
2.17 (a) Ligature pleine (en rouge), (b) Semi-ligature (en bleu). 50
2.18 (a) Squelette d'une vache mettant en évidence les ligatures, (b) Graphe d'os de la vache. 51
2.19 Illustration de la hiérarchie de Borgefors et al. (2001). 52
2.20 Illustration d'une séquence de squelettes hiérarchisée Yang et al. (2016). 52
2.21 Illustration des notations introduites pour calculer l'importance d'un point de squelette pour Shen et al. (2011). 53
2.22 (a) Squelette d'un chien, (b) squelette hiérarchique associé au chien, (c) schématisation de la hiérarchisation en couronne. 54
2.23 (a) Squelette d'une forme initiale de vache, (b) Squelette biaisé sur la forme de la vache légèrement lissée. 55
2.24 Illustration de la différence de structure du squelette de deux formes faisant partie de la même catégorie (Giang et al. (2013b)). 56

2.25 Trois opérations d'édition du graphe de chocs, (a) Jointure, (b) Contraction et (c) Fusion. 57
2.26 Illustration de l'appariement de deux DAGs. (a) Deux DAGs, (b) Création d'un graphe biparti associé à la pondération des arêtes suivant la valeur de similarité entre les nœuds, (c) Calcul de l'appariement maximum et ajout du meilleur à l'ensemble des solutions, (d) Scission des graphes au niveau des nœuds appariés, (e) Utilisation du même processus récursivement (Shokoufandeh et al. 2006). 58
2.27 Aperçu de la méthode de Demirci et al. (2006). 59
2.28 Graphe biparti complet dans lequel chaque sommet rouge est associé à la première forme et chaque sommet bleu, à la deuxième. Chaque sommet rouge est relié à chacun des sommets bleus. Le sommet bleu ciel est un sommet fantôme. . 60
2.29 (a) Squelette d'une feuille, (b) Squelette de cette même feuille privé de la zone saillante responsable de la branche turquoise. 61
2.30 Illustration du calcul du pageRank des pages web. Les sommets du graphe sont les pages web associées à leur pageRank et les arêtes orientées modélisent les hyperliens. 63

3.1 Diagramme de la méthode DECS 69
3.2 Représentation des boules maximales par des paraboloïdes elliptiques. 70
3.3 Coupe verticale représentant A,B,C, trois boules maximales. A,C appartiennent au squelette mais pas B car cette dernière est recouverte par l'union de A et de C, d'après Coeurjolly et Montanvert (2007) 71
3.4 *RDMA* d'une feuille de houx. 71
3.5 Carte des crêtes d'une feuille de houx résultant de la convolution de l'*EDT* avec un filtre *LoG* négatif. 72
3.6 Un exemple de résultat obtenu grâce à l'Algorithme de propagation. Les centres des boules maximales apparaissent en vert, les valeurs de la carte des crêtes supérieures ou égales à $th_{ridge-high}$ sont visibles en bleu et les valeurs de la carte des crêtes comprises entre $th_{ridge-low}$ et $th_{ridge-high}$ sont en rouge. 73
3.7 Un exemple de rares trous non désirés. 75
3.8 Un exemple de résultat obtenu grâce à l'algorithme d'amincissement MB2 que nous avons modifié pour n'avoir que des branches d'un pixel d'épaisseur. 76
3.9 Un exemple de squelette final obtenu avec la méthode DECS après élagage . . . 77
3.10 Squelette d'une lettre obtenu grâce à la méthode de Bertrand et Couprie (2014). . 78
3.11 Squelettes obtenus par la méthode de Choi et al. (2003) en utilisant différents seuils. 79
3.12 Champ de vecteurs généré par le calcul du gradient de la transformée en distance Euclidienne. Les vecteurs sont dirigés vers les crêtes de la carte de distance. Les points se situant à l'intersection des flux sont suceptibles d'être des points de squelette. Ils sont représentés par les lignes en pointillé rouge. 80

3.13 Squelettes obtenus grâce à la méthode de Siddiqi et al. (2002) en utilisant différents seuils. 80
3.14 Résultat de la squelettisation de forme bruitée pour chaque méthode avec le seuil le plus adapté (lorsqu'il existe), c'est-à-dire donnant la plus petite MHD, a) Méthode de Choi et al. (2003), b) Méthode de Siddiqi et al. (2002), c) Méthode de Bertrand et Couprie (2014) and d) Méthode DECS (Leborgne et al. 2015). . . . 82
3.15 Comparaison de la résistance au bruit. Pour i : indice du squelette $\in [1 \dots 15]$, pour t comme $th_{AOF} \in [-6 \dots -0,1]$ pour la méthode de Siddiqi et al. (2002), pour t comme $\rho \in [4 \dots 3000]$ pour la méthode de Choi et al. (2003) et t est la combinaison de $th_{perc-max-ball} \in [0 \dots 1]$ et $th_{ridge-high} \in [0,1 \dots 1,1]$ pour la méthode DECS. La MHD représentée est une moyenne des plus petites MHD obtenues sur chaque image. 83
3.16 Exemples de squelettes obtenus grâce à la méthode proposée en faisant varier les deux seuils. $th_{ridge-low}$ vaut toujours 0.05. 85
3.17 Évolution du seuil dans le but de mettre en évidence les déconnexions des squelettes obtenus avec la méthode de Choi et al. (2003). 86
3.18 Ces graphiques présentent la moyenne des temps de calcul de squelettes pour chaque image de la base de Latecki et al. (2000) en fonction de n, le nombre de pixels de l'image. Pour faire varier n, entre deux itérations, la hauteur et la largeur sont multipliées par $\sqrt{2}$. Par conséquent, n double à chaque itération. a) Temps de calcul total, b) Temps d'élagage (en rouge) et temps d'amincissement (en bleu). 87
3.19 (a) Exemple de squelette extrait à partir d'une feuille grâce à la méthode DECS ($th_{perc-max-ball}$=0.4 et $th_{ridge-high}$=0.5), (b) Image de différences obtenue. L'image reconstruite est représentée en gris et les pixels qui appartiennent à la forme d'origine mais pas à la forme reconstruite sont de couleur noire. 89
3.20 Variation des mesures Perc (a), HD (b) et MHD (c) en fonction des seuils $th_{perc-max-ball}$ et $th_{ridge-high}$ de DECS. 91
3.21 Variation des mesures Perc, HD et MHD en fonction du seuil th_{AOF} de la méthode de Siddiqi et al. (2002). Lorsque th_{AOF} diminue, les branches les moins importantes disparaissent. 92
3.22 Variation des mesures Perc, HD et MHD en fonction du seuil ρ de la méthode de Choi et al. (2003). Lorsque ρ augmente, les branches les moins importantes disparaissent. 92

4.1 Illustration des branches du squelette, qui sont plus ou moins importantes. (a) Squelette d'une feuille, (b) Squelette de cette même feuille avec quelques détails en plus. 101
4.2 Illustration de l'évolution de deux courbes sous l'effet du flot de courbure moyenne. 103

4.3 Illustration de l'évolution d'une courbe (contour de la forme de la Figure (a)) sous l'effet du flot de l'Équation 4.2. À chaque évolution, une forme est créée à partir de la courbe précédemment lissée. 104
4.4 Exemple de ré-échantillonnage du contour d'une forme. Le point bleu provient d'une fusion, alors que le point vert résulte d'une séparation. 106
4.5 Exemple du coloriage d'une forme à partir de son contour. 107
4.6 Illustration du vocabulaire utilisé pour les branches et les points de squelette. . . 107
4.7 Exemple d'appariement non trivial d'un point terminal de squelette entre deux étapes consécutives de lissage. 108
4.8 Exemple de réorganisation interne du squelette entre deux étapes consécutives de lissage. 109
4.9 Hiérarchisation du squelette : (a) squelette de la forme d'origine $S_{\mathcal{F}^{(0)}}$, (b) $S_{\mathcal{F}^{(6)}}$, suppression de la branche violette, (c) $S_{\mathcal{F}^{(15)}}$, suppression de la branche verte et fusion des branches bleue et turquoise, (d) $S_{\mathcal{F}^{(21)}}$, suppression de la branche jaune et fusion des branches rouge et orange, (e) $S_{\mathcal{F}^{(210)}}$, suppression de la branche turquoise et fusion des branches orange et rose, (f) $S_{\mathcal{F}^{(4015)}}$, réduction de la branche principale à un point. 111
4.10 Squelette hiérarchique S_h obtenu à partir de l'évolution représentée sur la Figure 4.9. Les étiquettes des méta-branches sont les nombres de lissages nécessaires pour la supprimer. 111
4.11 Squelette hiérarchique avant (a) et après (b) la mise à jour. $\boldsymbol{p}$ est l'endroit où les branches rose et verte sont concaténées. Les métabranches sont étiquetées avec leur poids. 112
4.12 Cas où une méta-branche interne devient une méta-branche terminale durant la mise à jour du squelette hiérarchique. Par conséquent, $\boldsymbol{p}$ devient un point terminal. Les méta-branches sont étiquetées avec leur poids. 112
4.13 Propagation des étiquettes après suppression d'une branche : la suppression du sommet **r** provoque la disparition de la branche d'étiquette 6. Les deux branches adjacentes (respectivement étiquetées 4 et 7) sont fusionnées et les étiquettes sont combinées dans la branche fusionnée. 113
4.14 Forces attirant les sommets vers les crêtes de la carte de distance euclidienne . . 113
4.15 Exemple montrant les étapes critiques du squelette déformable et du squelette hiérarchique associé. Les labels des méta-branches sont les nombres de lissages nécessaires pour supprimer les méta-branches. 117
4.16 Hyper-graphe hiérarchique construit à partir du squelette hiérarchique présenté sur la Figure 4.15(f). 119
4.17 (a) Squelette hiérarchique d'une feuille de houx, (b) Squelette hiérarchique de la même feuille de houx avec une rotation de 45 degrés. 121
4.18 Squelette hiérarchique d'une feuille de houx ayant pour résolution 203×291 (a), 304×436 (b). 122

4.19 Exemple de notre méthode d'élagage appliquée sur un squelette de Bertrand et Couprie (Bertrand et Couprie 2014). 122
4.20 Nombre de lissages nécessaires à la transformation du squelette en un point sur un échantillon de formes ayant chacune une aire différente. 123

5.1 Illustration de l'appariement des différentes parties d'une main. 127
5.2 Illustration de l'échantillonnage de deux hyper-sommets ainsi que des deux mises en correspondance possibles des points échantillonnés. 132
5.3 Exemple du plus court chemin entre deux méta-branches. 133
5.4 Illustration de l'EMD grâce à l'image des tas de terre et des trous. Les flèches pleines représentent le déplacement optimal des tas de terre vers les trous. 135
5.5 Fonction de pénalisation basée sur l'importance des hyper-sommets. 138
5.6 Squelettes hiérarchiques de deux formes utilisés pour illustrer l'Algorithme 15. . 141
5.7 Illustration de l'Algorithme 15 à partir de l'exemple des deux formes de la Figure 5.6. 142
5.8 Appariement des nœuds terminaux de deux squelettes selon Bai et Latecki (2008). 145
5.9 Contexte de forme (Belongie et al. (2002)). (a) et (b) Échantillonnage du bord de deux formes. (c) Diagramme d'histogrammes log-polaire. (d), (e) et (f) Exemples de contextes de forme marqués par les symboles $\circ$, $\diamond$ et $\triangleleft$ sur les figures (a) et (b). (g) Appariement des formes (a) et (b). 146
5.10 Base de données Kimia 216 (Sebastian et al. 2004). 148
5.11 Nombre de formes retrouvées en fonction du poids α. 149
5.12 Nombre de formes retrouvées en fonction du poids β. 150
5.13 Nombre de formes retrouvées en fonction du paramètre k. 150
5.14 Nombre de formes retrouvées, à chaque rang, en utilisant la méthode naïve et le squelette déformable. 151
5.15 Exemple de squelettes à différents seuils. 152
5.16 Nombre de formes retrouvées, à chaque rang, en utilisant trois couples de seuils dans la construction des squelettes grâce à la méthode DECS. Le squelette 1 a pour seuils $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.1$, le squelette 2 $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.2$ et le squelette 3 $th_{perc-max-ball} = 0.4$ et $th_{ridge-high} = 0.5$. 153
5.17 Méta-branches obtenues sur deux formes appartenant à la même catégorie mais dont l'agencement des branches internes est différent 154
5.18 Nombre de formes retrouvées, au premier rang, pour chaque espèce de feuilles. . 156
5.19 Appariement de deux feuilles d'arbres provenant de la même espèce (carica) et ayant des formes très différentes. La valeur de dissimilarité dans ce cas est 932. . 156
5.20 Appariement de deux feuilles d'arbres provenant de deux espèces différentes (grandiflora et aquifolium) mais ayant des formes très proches. La valeur de dissimilarité dans ce cas est 116. 157
5.21 Notations et illustration d'un graphe de Reeb sur une forme simple. 163

5.22 Squelette d'une spirale bruitée et son graphe de Reeb associé. Les pixels du squelette sont coloriés en fonction de leur valeur de h (voir palette) et les carrés coloriés de la même couleur représentent l'ensemble des pixels ayant la même valeur. 163
5.23 Extraction du graphe de Reeb à partir d'une image d'angiographie. (a) Squelette des vaisseaux sanguins sur une image d'angiographie segmentée. Les pixels du squelette sont coloriés en fonction de leur valeur de h (voir palette) et les carrés coloriés de la même couleur représentent l'ensemble des pixels ayant la même valeur. (b) Graphe de Reeb associé superposé à l'image initiale. 164
5.24 Travaux préliminaires mettant en évidence la structure minimale du squelette, en rose, comme un ensemble de méta-branches lorsqu'une forme présente un trou. 165

Introduction Générale

Au fur et à mesure de leur développement, les êtres humains acquièrent des compétences de plus en plus sophistiquées qui les aident à percevoir leur environnement et à prendre des décisions en fonction de leurs observations. Par exemple, il devient naturel de reconnaître un visage, de lire un manuscrit, de comprendre des mots parlés, de savoir si la nourriture est fraîche *etc.*

L'acquisition par une machine de ces capacités est un des principaux challenges en informatique. Pour cela, il faut réussir à lui retranscrire ce que l'être humain perçoit, la manière dont il analyse les choses et résout les problèmes. De nos jours, une machine est un appareil doué de grandes capacités de calculs mais qui s'adapte difficilement à une situation qui n'a pas été prévue dans sa programmation.

Pour simuler le comportement humain, nous appliquons de nombreuses techniques purement numériques n'ayant aucune correspondance avec les systèmes naturels : les algorithmes.

Pour prendre une décision, l'être humain se base sur son vécu et son environnement. Ainsi, il n'est pas obligé d'analyser tous les cas possibles, aussi simples soient-ils, pour réagir. Par exemple, s'il voit quelque chose qui sort de l'eau, il y a très peu de chances pour que cela soit un éléphant. *A priori*, l'être humain va penser à un poisson, voire un oiseau, avant d'y regarder de plus près. La machine n'a pas ces préjugés là donc, de manière naïve, elle va tester la majeure partie des solutions pour parvenir au meilleur résultat. Le temps de calcul pour prendre une décision devient donc rapidement conséquent si on ne trouve pas de ruses pour optimiser la méthode.

Dans cette thèse, nous allons nous intéresser spécifiquement à la reconnaissance des objets. Pour ce faire, nous considérons qu'une image est une capture du contenu visuel auquel il faut donner un sens. Plus précisément, nous voulons définir un degré de ressemblance entre plusieurs objets représentés dans différentes images afin de les reconnaître et les identifier. Pour réaliser cette tâche, l'être humain peut se baser sur les caractéristiques géométriques (particularités concernant la forme) ou sur des spécificités photométriques (intensité, couleur, texture). La forme est clairement le signal le plus important pour la reconnaissance, c'est un indice visuel significatif pour la perception humaine. En effet, les êtres humains sont souvent capables de reconnaître un objet uniquement sur la base de sa forme qu'ils décomposent en caractéristiques visuelles élémentaires. Pour illustrer ceci, nous prenons l'exemple d'un objet composé d'un long rectangle fin avec une multitude de rectangles d'une largeur minime et d'une longueur moyenne, perpendiculaires à l'extérieur d'un seul côté long de ce dernier et légèrement espacés entre eux (Figure 1). La plupart des gens vont associer cette forme à un peigne. Schomaker et al. (1999) montrent l'intérêt de l'utilisation des formes, par rapport aux couleurs et textures, pour reconnaître un objet. Bien sûr, si on précise que la couleur est noire et que la tex-

ture est un plastique brillant, la reconnaissance est plus facile. En revanche, si on cherche un objet noir en plastique brillant sans avoir d'indication sur la forme, c'est nettement plus compliqué car cela peut être une souris d'ordinateur, un stylo, une bouilloire *etc.* C'est la raison pour laquelle nous avons choisi de focaliser ce travail de thèse sur la forme. Nous avons, en effet, estimé que nous pouvions contribuer à la mise en place de nouveaux algorithmes dans ce domaine.

FIGURE 1 – *Représentation d'un peigne.*

La reconnaissance de formes est utilisée dans une multitude de domaines :
- la recherche d'images par le contenu visuel (Smeulders et al. 2000)
- navigation d'un robot (Ghazouani 2012)
- la reconnaissance d'identité (Maltoni et al. 2009, Bhanu et Tan 2012)
- la vidéo surveillance et suivi des objets en mouvement (Rougier et al. 2007)
- l'analyse d'images médicales et de protéines (Paragios et al. 2008)
- la reconnaissance de caractères (Surinta et al. 2015)
- *etc.*

La Figure 2 montre un schéma général sur lequel s'appuie la reconnaissance d'objets. Le fait de réduire un objet à une forme se trouve dans la partie saisie. Viennent ensuite les prétraitements qui permettent d'"améliorer" la forme brute en vue d'extraire différentes caractéristiques de la forme. En d'autres termes, il s'agit de segmenter une image pour en extraire la forme.

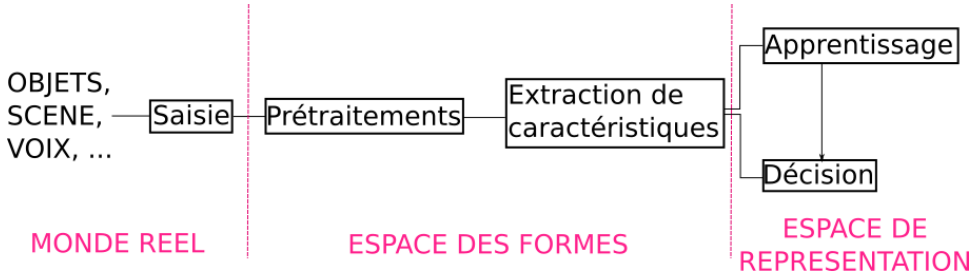


FIGURE 2 – *Schéma général d'un système de reconnaissance d'objets d'après Likforman-Sulem et Barney-Smith (2013).*

Pour reconnaître une forme, il est nécessaire d'apparier la forme "requête" aux différentes formes "type" (dont la classe associée est connue) pour déterminer celle qui s'en rapproche le plus et ainsi savoir de quel objet il s'agit (étape de décision). Le but des formes "type" est de servir de références, elles sont le résultat de l'étape d'apprentissage.

Une des applications de cette thèse est la reconnaissance de feuilles d'arbre (projet "Reconnaissance de Végétaux Récréative, Interactive et Educative sur Smartphone" ANR-15-CE38-0004). L'idée est de prendre en photo une feuille d'arbre (étape de saisie), puis d'isoler cette dernière de son contexte de manière à n'avoir que la forme dans l'image (étape de prétraitement/segmentation). Vient ensuite l'étape d'extraction de caractéristiques du contour de

la forme (Cerutti (2013)) ou de la forme dans sa globalité *via* des descripteurs topologiques par exemple. Dans cette thèse, nous nous intéresserons à ce dernier cas. Les formes "type" (diverses espèces de feuilles connues) ont été apprises (étape d'apprentissage) pour pouvoir apparier la forme "requête" à chacune d'elles et déterminer celle qui s'en rapproche le plus, donc son espèce (étape de décision).

L'objectif de cette thèse est de reconnaître des formes, en particulier des feuilles d'arbre, en développant un descripteur de forme qui permette de décrire aussi bien l'allure globale de la forme que les détails tout en maîtrisant la façon dont nous les utilisons lors de l'appariement.

Nos contributions portent sur l'extraction de caractéristiques ainsi que sur l'appariement de formes.

Le *premier chapitre* expose les notions de géométrie discrète et divers algorithmes connus dans le traitement d'images nécessaires pour la compréhension des chapitres suivants.

Le *deuxième chapitre* présente en détail la bibliographie portant sur toutes les étapes de la reconnaissance de formes planes, des descripteurs de forme à l'appariement de formes.

Le *troisième chapitre* montre la première étape de l'obtention de notre descripteur de forme, à savoir l'algorithme de squelettisation DECS.

Le *quatrième chapitre* décrit la pondération des branches en fonction de leur importance dans la forme de manière à obtenir un descripteur de forme plus riche que le squelette de base. Ce descripteur est modélisé par un hypergraphe pondéré et peut être utilisé aussi bien pour l'appariement de forme que pour l'élagage du squelette, en supprimant les branches les moins importantes.

Le *cinquième chapitre* expose la dernière étape de la reconnaissance de formes planes. Nous avons développé un algorithme d'appariement de formes que nous avons appliqué aux bases de données de la littérature ainsi que sur une base de données expérimentale composée de feuilles d'arbres.

Cette thèse a fait l'objet de plusieurs publications d'audiences nationales et internationales. Notre nouvel algorithme de squelettisation a été présenté lors d'une conférence internationale (International Symposium on Visual Computing) à Las Vegas (USA) en 2014 (Leborgne et al. 2014a) et dans une conférence nationale (COmpression et REprésentation des Signaux Audiovisuels) à Reims (France) en 2014 (Leborgne et al. 2014b). Nous avons ensuite proposé une version étendue au journal international (Journal of Visual Communication and Image Representation) en 2015 (Leborgne et al. 2015). Par ailleurs, nous allons présenter nos travaux portant sur la hiérarchisation dans une conférence internationale IEEE (International Conference on Image Processing) à Phoenix (USA) en septembre 2016 (Leborgne et al. 2016a) et dans une conférence nationale (Reconnaissance de Formes et Intelligence Artificielle) à Clermont-Ferrand en juin 2016 (Leborgne et al. 2016b).

1 Notions et algorithmes de base de géométrie discrète

Sommaire

1.1 Image numérique en deux dimensions . . . 7
1.2 Forme dans une image numérique . . . 8
1.3 Topologie et point simple . . . 10
1.3.1 Composante connexe . . . 11
1.3.2 Point simple . . . 12
1.3.3 Suivi de contours . . . 15
1.4 Segment discret entre deux pixels . . . 18
1.5 Distances discrètes et transformations en distance . . . 18
1.5.1 Transformée en Distance Euclidienne au Carré (*SEDT*) . . . 21
1.5.2 Transformée en Distance Euclidienne Séquentielle Signée . . . 25
1.6 Boules maximales discrètes et axe médian discret . . . 26
Conclusion . . . 29

Pour vous parler franchement de la Géométrie, je la trouve le plus haut exercice de l'Esprit."

PASCAL

La géométrie discrète est une branche de la géométrie, au même titre que la géométrie continue. Elle étudie les propriétés des objets (les courbes et les surfaces) définis par des ensembles de points discrets.

L'intérêt porté à la géométrie discrète s'est accru grâce aux progrès en matière d'imagerie. Depuis les années 60, des travaux, comme ceux de Rosenfeld et Pfaltz (1966), Rosenfeld (1970), ont permis de développer l'informatique graphique. De plus, durant ces dernières décennies, nous avons vu apparaître des appareils d'acquisition d'images multiples comme les appareils photos numériques, les scanners, les IRM *etc.* Tous ces appareils permettent de représenter l'information visuelle en ne manipulant que des données discrètes (*cf.* Figure 1.1). Par conséquent, il a fallu apprendre à analyser ces données, d'où le développement de la géométrie discrète.

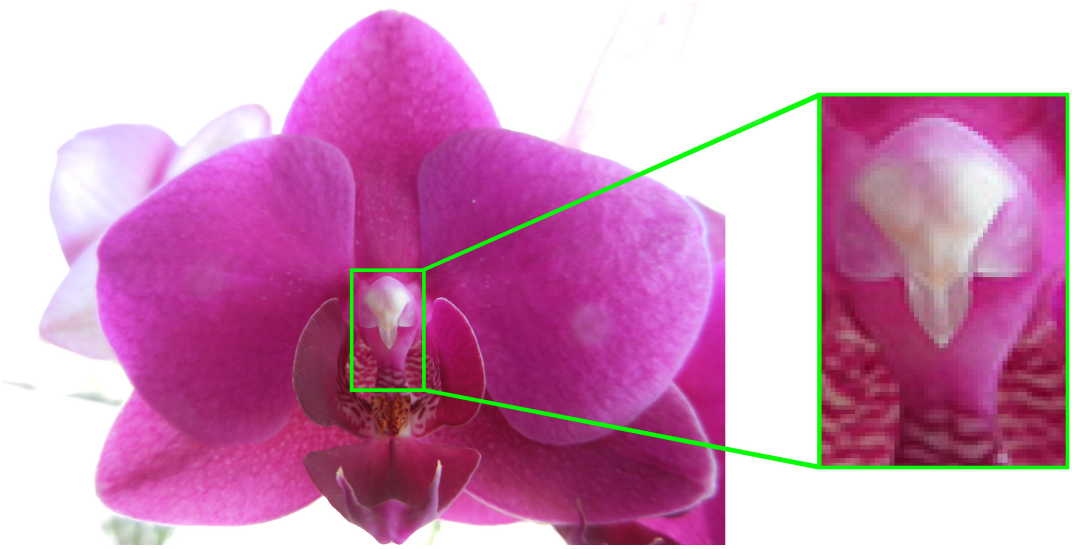


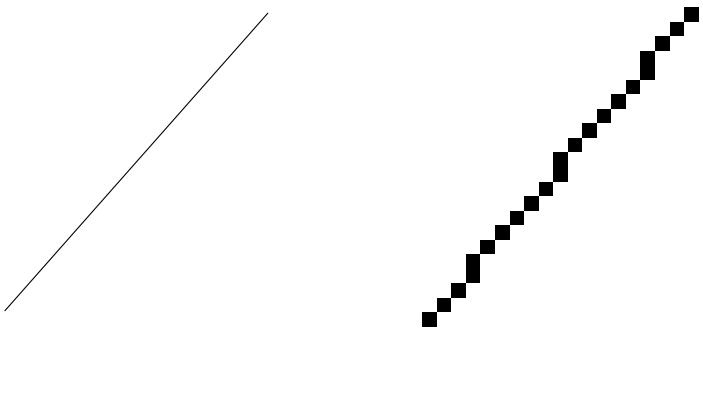
FIGURE 1.1 – *Mise en évidence de la discrétisation lors de l'utilisation d'un appareil photo numérique.*

Afin de bien comprendre la philosophie de ces deux géométries, nous allons prendre un exemple simple (*cf.* Figure 1.2). Plaçons-nous dans un espace en deux dimensions. En géométrie continue, un segment de droite est défini comme un ensemble infini de points alignés à coordonnées réelles. En revanche, en géométrie discrète, un segment de droite est formé par un ensemble fini de points à coordonnées entières.

Dans ce chapitre, nous introduisons les notions et algorithmes de base nécessaires à la compréhension de cette thèse en matière de géométrie discrète.

1.1 Image numérique en deux dimensions

Une image numérique $\mathcal{I} \subset \mathbb{Z}^2$ de taille $M \times N$ est un sous-ensemble du plan discret. Il s'agit d'une partition d'un sous-domaine rectangulaire de $\mathbb{R}^2$ en pavés élémentaires. Le pavage dans lequel nous allons travailler est un pavage carré régulier comme illustré sur la Figure 1.3. Chaque pavé élémentaire est associé à un point (également appelé *pixel*) $\boldsymbol{p}$ de coordonnées (x_p, y_p), qui est généralement son barycentre.



(a) (b)

FIGURE 1.2 – *Représentation d'un segment de droite (a) continu, (b) discret.*

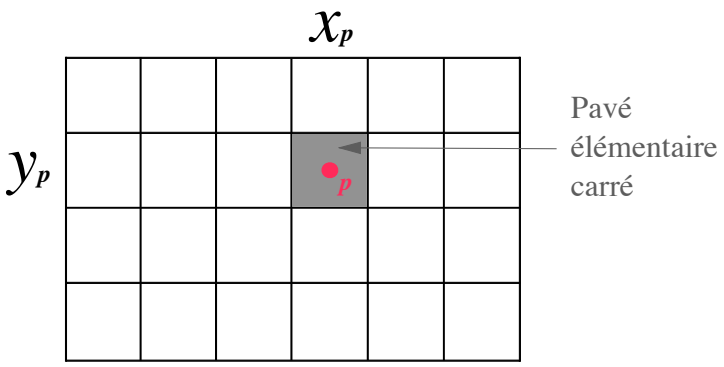


FIGURE 1.3 – *Illustration d'un pavage carré régulier permettant de représenter une image numérique.*

Dans le cadre de cette thèse, nous travaillerons sur des images binaires issues d'une segmentation.

1.2 FORME DANS UNE IMAGE NUMÉRIQUE

UNE forme $\mathcal{F}$ est un sous-ensemble de $\mathcal{I}$. $\mathcal{F}$ a une topologie qui lui est propre (cette notion sera abordée dans la partie suivante) et des relations particulières entre ses différents pixels.

Commençons par décomposer un pixel. D'après Kovalevsky (1990), chaque arête du carré élémentaire associée à un pixel est appelée un *lignel* et chaque sommet est appelé un *pointel*. Nommons les différents lignels ℓ^*_{pq}, où p et q sont les deux pixels qui ont ce lignel en commun. Les pointels sont nommés $\wp^*_{pqrs}$ où p, q, r et s sont les quatre pixels qui ont ce pointel en commun et qui sont ordonnés dans le sens des aiguilles d'une montre. Cette décomposition est illustrée sur la figure 1.4.

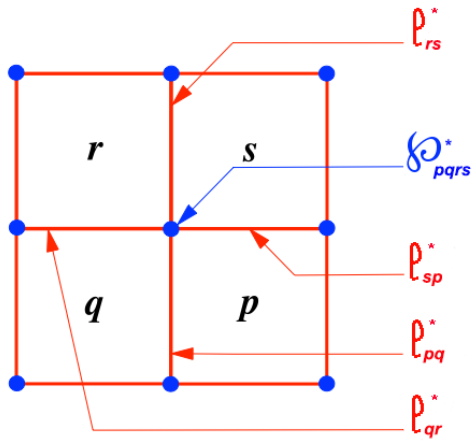
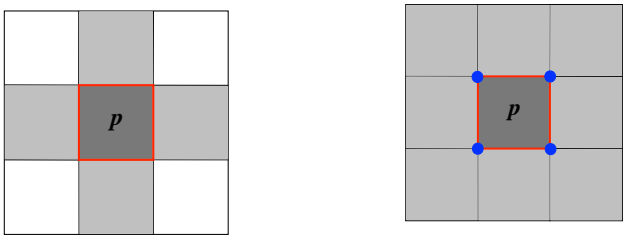


FIGURE 1.4 – *Décomposition d'un pixel. Les pointels sont en bleu et les lignels en rouge.*



(a) $\mathcal{N}_4(\boldsymbol{p})$ (b) $\mathcal{N}_8(\boldsymbol{p})$

FIGURE 1.5 – *(a) Voisinage 4-connexe, (b) Voisinage 8-connexe.*

Dans notre pavage carré régulier, nous allons considérer deux types de voisinage : le voisinage 8-connexe et le voisinage 4-connexe.

Définition 1.1 *Le **voisinage 8-connexe** $(\mathcal{N}_8(\boldsymbol{p}))$ de $\boldsymbol{p}$ ayant pour coordonnées (x_p, y_p) est l'ensemble des points $\boldsymbol{q}$ de coordonnées (x_q, y_q) défini par :*

$$\mathcal{N}_8(\boldsymbol{p}) = \{\boldsymbol{q} \mid \max(|x_p - x_q|, |y_p - y_q|) = 1\}$$

Définition 1.2 *Le **voisinage 4-connexe** $(\mathcal{N}_4(\boldsymbol{p}))$ de $\boldsymbol{p}$ ayant pour coordonnées (x_p, y_p) est l'ensemble des points $\boldsymbol{q}$ de coordonnées (x_q, y_q) défini par :*

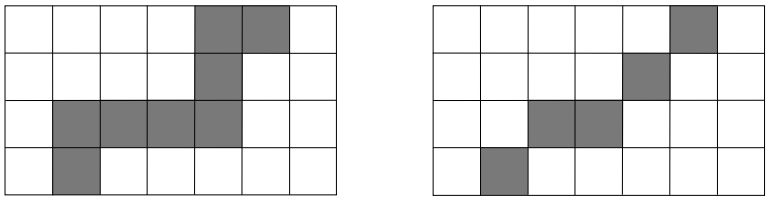
$$\mathcal{N}_4(\boldsymbol{p}) = \{\boldsymbol{q} \mid |x_p - x_q| + |y_p - y_q| = 1\}$$

Un point $\boldsymbol{q}$ appartient au voisinage 4-connexe (respectivement 8-connexe) de $\boldsymbol{p}$ si $\boldsymbol{p}$ et $\boldsymbol{q}$ ont un lignel en commun (respectivement un pointel en commun ou un lignel en commun) comme le montre la Figure 1.5.

Pour définir une forme, nous avons besoin d'une dernière notion appelée chemin.

Définition 1.3 *Un chemin 4-connexe (respectivement, 8-connexe) est une séquence $(\boldsymbol{p}_0, ..., \boldsymbol{p}_{u-1})$ de u pixels de $\mathbb{Z}^2$ tels que pour tout $i = 0, ..., (u-2)$, $\boldsymbol{p}_{i+1}$ appartient à $\mathcal{N}_4(\boldsymbol{p}_i)$ (respectivement, $\mathcal{N}_8(\boldsymbol{p}_i)$).*

Cette définition est illustrée par la Figure 1.6. On notera qu'un chemin 4-connexe est également 8-connexe.



(a) Chemin 4-connexe (b) Chemin 8-connexe

FIGURE 1.6 – *Exemples de chemins 4 et 8 connexes.*

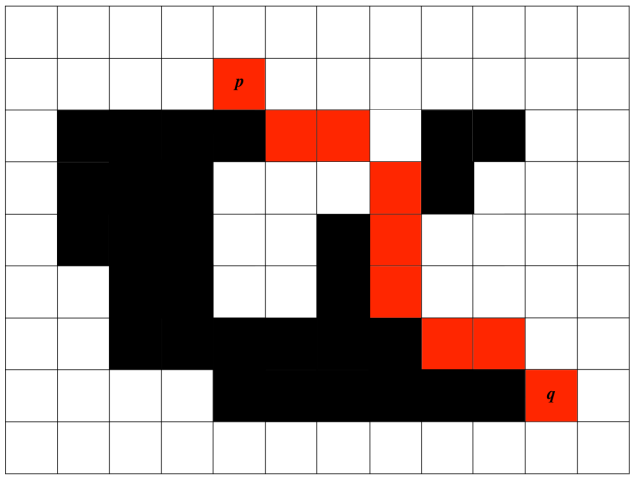


FIGURE 1.7 – *Illustration de la notion de forme 8-connexe.*

À ce stade, nous sommes en possession de toutes les notions nécessaires pour définir une forme.

Définition 1.4 *Une forme $\mathcal{F}$ est un ensemble de pixels de $\mathbb{Z}^2$ satisfaisant une propriété de connexité. $\mathcal{F}$ est une forme, si et seulement si, pour chacune des paires de pixels $\boldsymbol{p}$ et $\boldsymbol{q}$ appartenant à $\mathcal{F}$, il existe un chemin 8-connexe entre $\boldsymbol{p}$ et $\boldsymbol{q}$ inclus dans $\mathcal{F}$. Nous appellerons $\bar{\mathcal{F}}$ le complémentaire de $\mathcal{F}$ dans $\mathcal{I}$, autrement dit, le fond de l'image.*

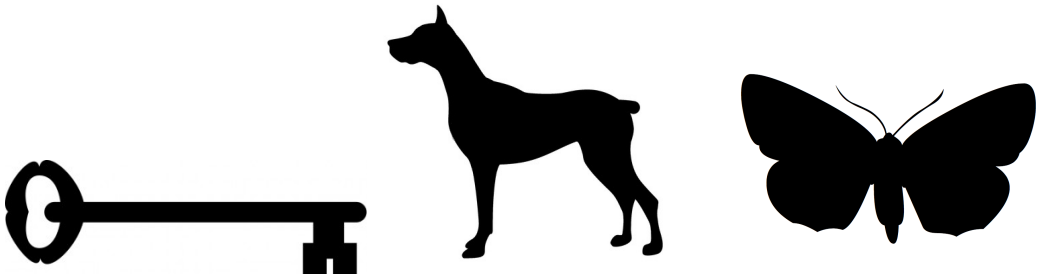
Dans la suite de cette thèse, nous ne travaillerons que sur des formes 8-connexes, ce qui implique que le fond soit 4-connexe afin de vérifier le théorème de Jordan (Rosenfeld 1973).

La Figure 1.7 permet d'illustrer la Définition 1.4. Les pixels de $\mathcal{F}$ sont représentés en noir et en rouge, les pixels de $\bar{\mathcal{F}}$ en blanc. Les pixels représentés en rouge mettent en évidence un chemin 8-connexe entre $\boldsymbol{p}$ et $\boldsymbol{q}$ inclus dans $\mathcal{F}$.

La Définition 1.4 implique qu'une forme peut être trouée ou non. Donc, deux formes peuvent avoir des topologies différentes.

1.3 TOPOLOGIE ET POINT SIMPLE

LA topologie est une discipline appartenant aux mathématiques. Dans le cadre de cette thèse, nous nous intéressons à l'équivalence topologique entre deux formes. En d'autres



(a) (b) (c)

FIGURE 1.8 – *Exemples de formes binaire. (a) clé, (b) chien et (c) papillon.*

mots, nous cherchons à transformer une forme en une autre sans utiliser de déformations telles que l'arrachage et le recollement.

Prenons trois exemples afin de clarifier les choses. Sur la Figure 1.8, sont représentées trois formes binaires (une clé, un chien et un papillon). Le papillon et le chien sont topologiquement équivalents puisqu'en utilisant des rotations, translations, homothéties *etc.* nous pouvons obtenir le papillon à partir du chien et *vice versa*. En revanche, la clé n'est pas topologiquement équivalente au papillon ni au chien car elle possède un trou, contrairement aux deux autres formes. Pour créer ce trou, sur le chien par exemple, il faudrait arracher une partie de la forme. La Définition 1.6 spécifie la notion de trou formellement.

Dans cette thèse, nous allons avoir besoin de deux notions en relation directe avec la topologie. Il s'agit des notions de *composante connexe* et *point simple*.

1.3.1 Composante connexe

Comme précisé dans la Partie 1.2, nous travaillons avec des formes 8-connexes sur un fond 4-connexe durant toute cette thèse.

Définition 1.5 *Une composante 4-connexe (respectivement, 8-connexe) est un ensemble maximal $\mathcal{P}$ de pixels tel que pour tous pixels $\boldsymbol{p}_i$ et $\boldsymbol{p}_j$ de $\mathcal{P}$, il existe un chemin 4-connexe (respectivement, 8-connexe) dans $\mathcal{P}$.*

La Figure 1.9 permet d'illustrer les notions de composantes 4-connexes et 8-connexes. Les deux points noirs font partie de la même composante 8-connexe puisque le pixel $\boldsymbol{q}$ appartient au voisinage 8-connexe du pixel $\boldsymbol{p}$. En revanche, les pixels $\boldsymbol{p}^{\triangleright}$ et $\boldsymbol{q}^{\triangleright}$ ne font pas partie de la même composante 4-connexe puisque $\boldsymbol{q}^{\triangleright}$ n'appartient pas au voisinage 4-connexe de $\boldsymbol{p}^{\triangleright}$. Donc, dans cet exemple, les pixels noirs forment une seule composante 8-connexe et les pixels blancs deux composantes 4-connexes.

Il est à noter que, d'après la Définition 1.4, l'ensemble des pixels noirs appartient à une seule et même forme.

Grâce à la notion de composante connexe, nous pouvons définir ce qu'est un trou.

Définition 1.6 *Une forme $\mathcal{F}$, dont aucun pixel n'est présent sur le bord de l'image, comporte u trous*

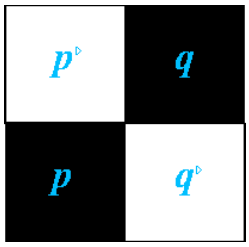


FIGURE 1.9 – *Illustration de la notion de composante 4-connexe et 8-connexe.*

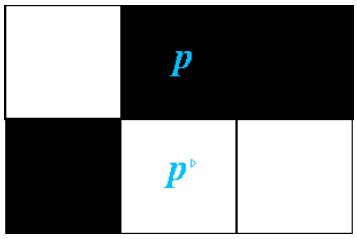


FIGURE 1.10 – *Illustration de la notion de point simple.*

si $\bar{\mathcal{F}}$ est constituée de $u+1$ composantes connexes. En d'autres termes, une forme est une composante 8-connexe.

1.3.2 Point simple

La notion de point simple est beaucoup utilisée en squelettisation, en particulier pour caractériser la conservation de la topologie.

Définition 1.7 *Point simple : Soit $\mathcal{P}$ un sous-ensemble de $\mathbb{Z}^2$ avec un nombre fini de composantes 8-connexes correspondant à $\mathcal{F}$ et de composantes 4-connexes correspondant à $\bar{\mathcal{F}}$. Un point $\boldsymbol{p}$ de $\mathcal{P}$ est 4-simple (respectivement, 8-simple) dans $\mathcal{P}$ si :*

- *les ensembles $\mathcal{P}$ et $\mathcal{P} \setminus \{\boldsymbol{p}\}$ ont le même nombre de composantes 4-connexes (respectivement, 8-connexes).*
- *les ensembles $\overline{\mathcal{P}}$ et $\overline{\mathcal{P} \setminus \{\boldsymbol{p}\}}$ ont le même nombre de composantes 8-connexes (respectivement, 4-connexes).*

En d'autres termes, un point de fond est simple si sa transformation en un point de forme, et *vice versa*, ne modifie ni le nombre de composantes connexes du fond, ni le nombre de composantes connexes de la forme.

La Figure 1.10 illustre cette notion. $\boldsymbol{p}$ n'est pas un point 8-simple car sa transformation en un pixel appartenant à $\bar{\mathcal{F}}$ impliquerait que les pixels noirs forment deux composantes connexes distinctes et les pixels blancs une seule composante connexe. En revanche, $\boldsymbol{p}^{\triangleright}$ est un point simple.

Dans la pratique, pour savoir si un point est simple, nous utilisons un critère calculable sur son voisinage. Ce critère prend différents noms en fonction des auteurs (nombre d'intersections, nombre de connectivités, *etc.*).

Pour cela, nous utilisons la fonction indicatrice d'un sous-ensemble $\mathcal{F}$ de $\mathcal{I}$ pour un point

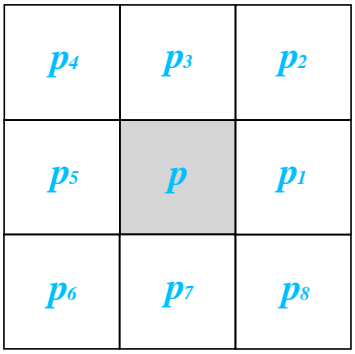


FIGURE 1.11 – *Indexation du voisinage d'un pixel.*

$\boldsymbol{p} \in \mathcal{I}$ notée $\mathbf{1}_{\mathcal{F}}$ et définie par :

$$\begin{array}{rcl} \mathbf{1}_{\mathcal{F}} : I & \longrightarrow & \{0,1\} \\ \boldsymbol{p} & \longmapsto & \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}) = \begin{cases} 1 \text{ si } \boldsymbol{p} \in \mathcal{F} \\ 0 \text{ si } \boldsymbol{p} \in \bar{\mathcal{F}} \end{cases} \end{array} \tag{1.1}$$

Par convention, les voisins de $\boldsymbol{p}$ sont ordonnés comme illustré sur la Figure 1.11.

Le tout premier critère a été défini par Rutovitz (1966).

Définition 1.8 *Nombre d'intersections de RUTOVITZ :*

$$X_R(\boldsymbol{p}) = \sum_{i=1}^{8} \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_{i+1}) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_i) \mid, \textit{avec } \boldsymbol{p}_9 = \boldsymbol{p}_1$$

Le point $\boldsymbol{p}$ est simple si et seulement si $X_R(\boldsymbol{p}) = 2$. En d'autres termes, $X_R(\boldsymbol{p})$ représente le nombre de transitions entre les pixels noirs et blancs et *vice versa*.

Le critère de Rutowitz est basé sur la 4-connexité. Ainsi, les points 8-simples mais non 4-simples ne seront pas détectés, comme le montre l'exemple de la Figure 1.12. Le point $\boldsymbol{p}$ est 8-simple mais non 4-simple. En effet, $X_R(\boldsymbol{p}) = 4$ et pourtant le point est 8-simple.

$$\begin{aligned} X_R(\boldsymbol{p}) = & \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_2) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_1) \mid + \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_3) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_2) \mid + \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_4) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_3) \mid + \\ & \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_5) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_4) \mid + \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_6) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_5) \mid + \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_7) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_6) \mid + \\ & \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_8) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_7) \mid + \mid \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_1) - \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_8) \mid \\ X_R(\boldsymbol{p}) = & \, 0+1+1+1+1+0+0 \\ X_R(\boldsymbol{p}) = & \, 4 \end{aligned}$$

Pour résoudre ce problème, trois années plus tard, Hilditch (1969) a introduit un nombre d'intersections prenant en compte la 8-connexité :

Définition 1.9 *Nombre d'intersections de HILDITCH :*

$$X_H(\boldsymbol{p}) = \sum_{i=1}^{4} b_i$$

avec

$$b_i = \begin{cases} 1 \text{ si } \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_{2i-1}) = 0 \text{ et } (\mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_{2i}) = 1 \text{ ou } \mathbf{1}_{\mathcal{F}}(\boldsymbol{p}_{2i+1}) = 1) \\ 0 \text{ sinon} \end{cases} \quad \text{et } \boldsymbol{p}_9 = \boldsymbol{p}_1$$

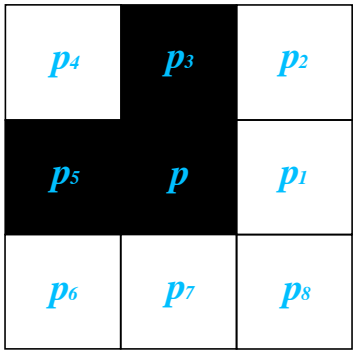


FIGURE 1.12 – *Exemple d'un point p 8-simple mais non 4-simple.*

Si $X_H(p) = 1$, alors p est 8-simple. La Figure 1.12 permet d'illustrer la Définition 1.9 dans laquelle p est 8-simple. Le calcul du critère de HILDITCH est :

$$\begin{aligned} X_H(p) &= b_1 + b_2 + b_3 + b_4 \\ X_H(p) &= 1 + 0 + 0 + 0 \\ X_H(p) &= 1 \end{aligned}$$

Le problème de ce critère est qu'on ne peut pas déterminer si un point est 8-simple mais non 4-simple. Pour résumer, nous avons le critère de RUTOVITZ basé sur la 4-connexité, et le critère de HILDITCH basé sur la 8-connexité. Pour homogénéiser ces deux critères, Yokoi et al. (1975) a proposé les nombres de connexité.

Définition 1.10 *Nombres de connexité de YOKOI :*

$$Y_4(p) = \sum_{i=1}^{4} (\mathbf{1}_{\mathcal{F}}(p_{2i-1}) - \mathbf{1}_{\mathcal{F}}(p_{2i-1}) \times \mathbf{1}_{\mathcal{F}}(p_{2i}) \times \mathbf{1}_{\mathcal{F}}(p_{2i+1}))$$

$$Y_8(p) = \sum_{i=1}^{4} (\overline{\mathbf{1}_{\mathcal{F}}(p_{2i-1})} - \overline{\mathbf{1}_{\mathcal{F}}(p_{2i-1})} \times \overline{\mathbf{1}_{\mathcal{F}}(p_{2i})} \times \overline{\mathbf{1}_{\mathcal{F}}(p_{2i+1})})$$

avec $p_9 = p_1$ *et* $\overline{\mathbf{1}_{\mathcal{F}}(p)} = 1 - \mathbf{1}_{\mathcal{F}}(p)$

Si $Y_4(p)$=1, alors p est 4-simple et si $Y_8(p)$=1, alors p est 8-simple. L'exemple de la Figure 1.12 permet d'illustrer la Définition 1.10.

$$\begin{aligned} Y_4(p) &= \mathbf{1}_{\mathcal{F}}(p_1) - \mathbf{1}_{\mathcal{F}}(p_1) \times \mathbf{1}_{\mathcal{F}}(p_2) \times \mathbf{1}_{\mathcal{F}}(p_3) + \mathbf{1}_{\mathcal{F}}(p_3) - \mathbf{1}_{\mathcal{F}}(p_3) \times \mathbf{1}_{\mathcal{F}}(p_4) \times \mathbf{1}_{\mathcal{F}}(p_5) + \\ &\quad \mathbf{1}_{\mathcal{F}}(p_5) - \mathbf{1}_{\mathcal{F}}(p_5) \times \mathbf{1}_{\mathcal{F}}(p_6) \times \mathbf{1}_{\mathcal{F}}(p_7) + \mathbf{1}_{\mathcal{F}}(p_7) - \mathbf{1}_{\mathcal{F}}(p_7) \times \mathbf{1}_{\mathcal{F}}(p_8) \times \mathbf{1}_{\mathcal{F}}(p_1) \\ Y_4(p) &= 0 - 0 \times 0 \times 1 + 1 - 1 \times 0 \times 1 + 1 - 1 \times 0 \times 0 + 0 - 0 \times 0 \times 0 \\ Y_4(p) &= 2 \end{aligned}$$

Donc p n'est pas 4-simple.

$$\begin{aligned} Y_8(p) &= \overline{\mathbf{1}_{\mathcal{F}}(p_1)} - \overline{\mathbf{1}_{\mathcal{F}}(p_1)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_2)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_3)} + \overline{\mathbf{1}_{\mathcal{F}}(p_3)} - \overline{\mathbf{1}_{\mathcal{F}}(p_3)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_4)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_5)} + \\ &\quad \overline{\mathbf{1}_{\mathcal{F}}(p_5)} - \overline{\mathbf{1}_{\mathcal{F}}(p_5)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_6)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_7)} + \overline{\mathbf{1}_{\mathcal{F}}(p_7)} - \overline{\mathbf{1}_{\mathcal{F}}(p_7)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_8)} \times \overline{\mathbf{1}_{\mathcal{F}}(p_1)} \\ Y_4(p) &= 1 - 1 \times 1 \times 0 + 0 - 0 \times 1 \times 0 + 0 - 0 \times 1 \times 1 + 1 - 1 \times 1 \times 1 \\ Y_4(p) &= 1 \end{aligned}$$

Donc p est 8-simple.

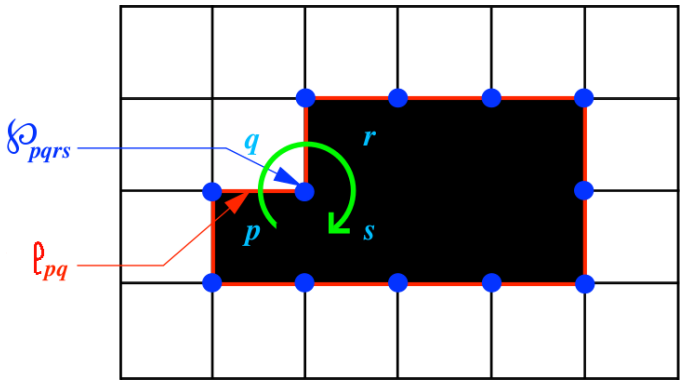


FIGURE 1.13 – *Exemple de pointel de contour et de lignel de contour.*

1.3.3 Suivi de contours

Travailler à partir de formes binaires discrètes requiert l'utilisation d'algorithmes propres à la géométrie discrète, tels que l'algorithme du suivi de contours. Ce dernier permet, comme son nom l'indique, d'obtenir le contour de la forme. Cette partie est inspirée de la Thèse de Roussillon (2009), qui lui-même s'est inspiré des travaux de Rosenfeld (1970) et de Kovalevsky (2001). Ce contour peut être une liste de lignels ou une liste de pointels, selon les besoins.

Définition 1.11 *Un lignel de contour est un lignel commun à un pixel $\boldsymbol{p} \in \mathcal{F}$ et à un pixel $\boldsymbol{q} \in \bar{\mathcal{F}}$. Ce lignel est noté $\ell_{\boldsymbol{pq}}$.*

Définition 1.12 *Un pointel de contour est un pointel commun à quatre pixels $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ et $\boldsymbol{s}$ tels que $\boldsymbol{p} \in \mathcal{F}$, $\boldsymbol{q} \in \bar{\mathcal{F}}$, $\boldsymbol{r}$ et $\boldsymbol{s}$ peuvent appartenir ou non à $\mathcal{F}$. $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ et $\boldsymbol{s}$ sont les sommets d'un carré unité ordonnés dans le sens horaire. Ces quatre pixels forment un quatromino (domino 2×2). Ce pointel est noté $\wp_{\boldsymbol{pqrs}}$.*

Les deux définitions ci-dessus sont illustrées par la Figure 1.13.

Pour déterminer le contour d'une forme, l'algorithme comporte deux étapes. La première est la recherche d'un point de départ et la seconde est le parcours du bord de la forme à partir du point de départ jusqu'à retrouver ce dernier.

Pour rechercher un point de départ (l. 2 à 15 de l'Algorithme 1), nous allons parcourir les lignes et les colonnes une à une jusqu'à trouver un motif précis composé de pixels appartenant au fond et à la forme. Comme nous supposons que la forme ne touche pas le bord de l'image, le motif recherché, est tel qu'un pixel $\boldsymbol{q}$ appartenant à $\bar{\mathcal{F}}$ est au-dessus d'un pixel $\boldsymbol{p}$ appartenant à $\mathcal{F}$, d'où la Figure 1.14.

La deuxième étape consiste à chercher le pointel suivant selon les directions de Freeman (Freeman 1961) en tournant toujours dans le même sens (par exemple horaire) jusqu'à revenir au pointel de départ. Cela revient à déplacer, en autorisant la rotation, le quatromino $\boldsymbol{pqrs}$ sur l'image en faisant correspondre les pixels $\boldsymbol{p}$ et $\boldsymbol{q}$ de l'image avec les pixels $\boldsymbol{p}$ et $\boldsymbol{q}$ du quatromino. En conséquence, nous pouvons déterminer la position des pixels $\boldsymbol{r}$ et $\boldsymbol{s}$ dans l'image (*cf.* Figure 1.15) et ajouter le lignel ℓ_{pq} à la liste de lignels $\mathcal{L}_\ell$ et le pointel $\wp_{pqrs}$ à la liste de pointels $\mathcal{L}_\wp$ (l. 19 à 21 de l'Algorithme 1).

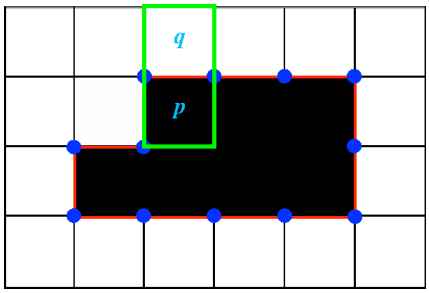


FIGURE 1.14 – *Motif de départ du suivi de contour.*

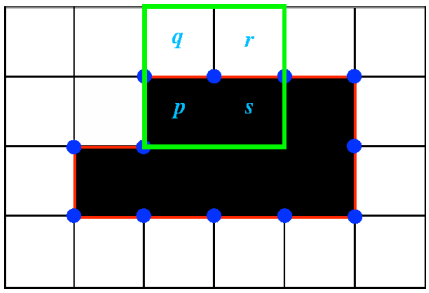
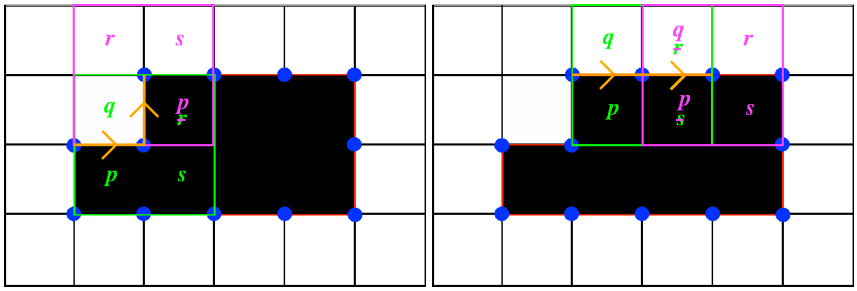
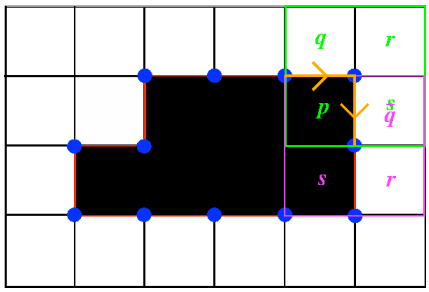


FIGURE 1.15 – *Placement du quatromino sur l'image.*



(a) Si $\boldsymbol{r} \in \mathcal{F}$ ⇒tourner à gauche (b) Si $\boldsymbol{r} \in \bar{\mathcal{F}}$ *et* $\boldsymbol{s} \in \mathcal{F}$ ⇒aller tout droit



(c) Si $\boldsymbol{r} \in \bar{\mathcal{F}}$ *et* $\boldsymbol{s} \in \bar{\mathcal{F}}$ ⇒tourner à droite

FIGURE 1.16 – *Illustration des différentes configurations du suivi de contour.*

La suite de l'algorithme consiste à déterminer les nouvelles coordonnées des pixels $\boldsymbol{p}$ et $\boldsymbol{q}$ en fonction de l'appartenance ou non des pixels $\boldsymbol{r}$ et $\boldsymbol{s}$ à $\mathcal{F}$ (l. 22 à 34 de l'Algorithme 1). Les trois configurations possibles sont illustrées par la Figure 1.16.

Algorithme 1: Suivi de contours.

Entrées : Image $\mathcal{I}$ de taille $M \times N$ et forme $\mathcal{F}$

Sorties : liste de pointels $\mathcal{L}_\wp$, liste de lignels $\mathcal{L}_\ell$

Hypothèse : Aucun pixel de $\mathcal{F}$ n'appartient au bord de l'image.

début
  */*Recherche d'un motif de départ*/*
  $i \leftarrow 1$; $stop \leftarrow faux$;
  tant que $(i < M-1)$ *et* $(\neg stop)$ **faire**
    $j \leftarrow 1$;
    tant que $(j < N-1)$ *et* $(\neg stop)$ **faire**
      si $(i,j) \in \mathcal{F}$ *et* $(i,j-1) \in \bar{\mathcal{F}}$ **alors**
        $\boldsymbol{p} \leftarrow (i,j)$;
        $\boldsymbol{q} \leftarrow (i,j-1)$;
        $stop \leftarrow vrai$;
      fin
      $j \leftarrow j+1$;
    fin
    $i \leftarrow i+1$;
  fin
  */*Suivi de contours*/*
  $\mathcal{L}_\ell \leftarrow \varnothing$; $\mathcal{L}_\wp \leftarrow \varnothing$; $\boldsymbol{p}_{initial} \leftarrow \boldsymbol{p}$; $\boldsymbol{q}_{initial} \leftarrow \boldsymbol{q}$;
  répéter
    $\boldsymbol{r} \leftarrow (x_q + y_q - y_p\,,\, y_q + x_p - x_q)$;
    $\boldsymbol{s} \leftarrow (x_p + y_q - y_p\,,\, y_p + x_p - x_q)$;
    $\mathcal{L}_\wp \leftarrow \mathcal{L}_\wp + \wp_{\boldsymbol{pqrs}}$; $\mathcal{L}_\ell \leftarrow \mathcal{L}_\ell + \ell_{\boldsymbol{pq}}$;
    */*aller à gauche*/*
    si $\boldsymbol{r} \in \mathcal{F}$ **alors**
      $\boldsymbol{p} \leftarrow \boldsymbol{r}$;
    fin
    */*aller tout droit*/*
    Sinon si $\boldsymbol{r} \in \bar{\mathcal{F}}$ *et* $\boldsymbol{s} \in \mathcal{F}$ **alors**
      $\boldsymbol{p} \leftarrow \boldsymbol{s}$;
      $\boldsymbol{q} \leftarrow \boldsymbol{r}$;
    fin
    */*aller à droite*/*
    Sinon si $\boldsymbol{r} \in \bar{\mathcal{F}}$ *et* $\boldsymbol{s} \in \bar{\mathcal{F}}$ **alors**
      $\boldsymbol{q} \leftarrow \boldsymbol{s}$;
    fin
  jusqu'à $(\boldsymbol{p}_{initial} = \boldsymbol{p})$ *et* $(\boldsymbol{q}_{initial} = \boldsymbol{q})$;
  retourner $\mathcal{L}_\wp$ *et* $\mathcal{L}_\ell$
fin

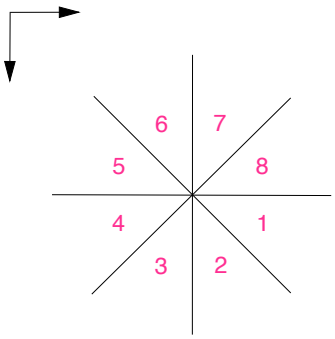


FIGURE 1.17 – *Décomposition de l'espace 2D en octants.*

1.4 SEGMENT DISCRET ENTRE DEUX PIXELS

POUR calculer le segment discret entre les deux points discrets $\boldsymbol{q}_i$ et $\boldsymbol{q}_{i+1}$, nous proposons d'utiliser la discrétisation *Grid Intersect Quantization* (GIQ), qui approxime le segment continu par l'ensemble des pixels les plus proches. Nous avons utilisé l'algorithme de Bresenham (1965). Pour cela, nous avons besoin de la notion d'octant pour un segment de droite (*cf.* Figure 1.17).

De manière plus formelle, dire que $\boldsymbol{q}_i$ et $\boldsymbol{q}_{i+1}$ se situent dans le premier octant revient à :

$$x_{q_{i+1}} - x_{q_i} > y_{q_{i+1}} - y_{q_i} > 0$$

Pour présenter l'algorithme de Bresenham, nous allons nous placer dans le premier octant. Ensuite, nous utiliserons des transformations permettant la généralisation du tracé dans les autres octants.

L'idée générale de cet algorithme est de tracer, pixel par pixel, un chemin 8-connexe entre $\boldsymbol{q}_i$ et $\boldsymbol{q}_{i+1}$. Comme nous sommes dans le premier octant, à chaque étape, nous avons deux possibilités pour tracer le pixel $\boldsymbol{q}_{u+1}$ $(x_{q_u} + 1, y_{q_u})$ se situant soit à l'est de $\boldsymbol{q}_u$, c'est-à-dire permettant de rester sur le même palier que $\boldsymbol{q}_u$, soit au sud-est de $\boldsymbol{q}_u$: $(x_{q_u} + 1, y_{q_u} + 1)$, en d'autres termes permettant de changer de palier par rapport à $\boldsymbol{q}_u$. Le choix se fait en propageant, à chaque itération, une mesure d'erreur de la position du pixel courant par rapport au segment réel. Lorsque l'erreur devient trop grande, on change de palier (pixel au sud-est), sinon on reste sur le même (pixel à l'est). L'algorithme de Bresenham est présenté dans l'Algorithme 2 et illustré par la Figure 1.18.

1.5 DISTANCES DISCRÈTES ET TRANSFORMATIONS EN DISTANCE

TOUT comme dans l'espace continu en deux dimensions, la mesure d'une distance entre deux points est possible dans l'espace discret en deux dimensions. Nous parlerons alors de distances discrètes.

Définition 1.13 *Distance discrète : Soit une fonction* $d : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}^+$ *et* $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}$ *des points appartenant à* $\mathbb{Z}^2$. *d est une distance si les trois conditions ci-dessous sont respectées :*

Algorithme 2: Algorithme de Bresenham : représentation d'un segment discret.

Entrées : Deux pixels : $q_i(x_{q_i}, y_{q_i})$ et $q_{i+1}(x_{q_{i+1}}, y_{q_{i+1}})$, octant.

Sorties : Segment discret entre q_i et q_{i+1}

```
1  début
2      suivant octant faire
3          cas où octant=1 x_{q_{i+1}} ← x_{q_{i+1}} ; y_{q_{i+1}} ← y_{q_{i+1}} cas où octant=2 x_{q_{i+1}} ← y_{q_{i+1}} ;
           y_{q_{i+1}} ← x_{q_{i+1}} cas où octant=3 x_{q_{i+1}} ← −y_{q_{i+1}} ; y_{q_{i+1}} ← x_{q_{i+1}} cas où octant=4
           x_{q_{i+1}} ← −x_{q_{i+1}} ; y_{q_{i+1}} ← y_{q_{i+1}} cas où octant=5 x_{q_{i+1}} ← −x_{q_{i+1}} ; y_{q_{i+1}} ← −y_{q_{i+1}} cas
           où octant=6 x_{q_{i+1}} ← −y_{q_{i+1}} ; y_{q_{i+1}} ← −x_{q_{i+1}} cas où octant=7 x_{q_{i+1}} ← y_{q_{i+1}} ;
           y_{q_{i+1}} ← −x_{q_{i+1}} cas où octant=8 x_{q_{i+1}} ← x_{q_{i+1}} ; y_{q_{i+1}} ← −y_{q_{i+1}}
4      fin
5      d_x ← x_{q_{i+1}} − x_{q_i} ; d_y ← y_{q_{i+1}} − y_{q_i} ;
6      x ← x_{q_i} ; y ← y_{q_i} ;
7      incrHor ← 2 × d_y ; incrDiag ← 2 × (d_y − d_x) ;
8      err ← 2 × d_y − d_x ;
9      pour u allant de x_{q_i} x_{q_{i+1}} faire
10         suivant octant faire
11             cas où octant=1 afficherPixel(x, y) cas où octant=2 afficherPixel(y, x) cas où
               octant=3 afficherPixel(−y, x) cas où octant=4 afficherPixel(−x, y) cas où
               octant=5 afficherPixel(−x, −y) cas où octant=6 afficherPixel(−y, −x) cas où
               octant=7 afficherPixel(y, −x) cas où octant=8 afficherPixel(x, −y)
12         fin
13         si err≥0 alors
14             /* Changement de palier*/
15             y=y+1;
16             err←err+incrDiag;
17         sinon
18             /* Même palier palier*/
19             err←err+incrHor;
20         fin
21     fin
22 fin
```

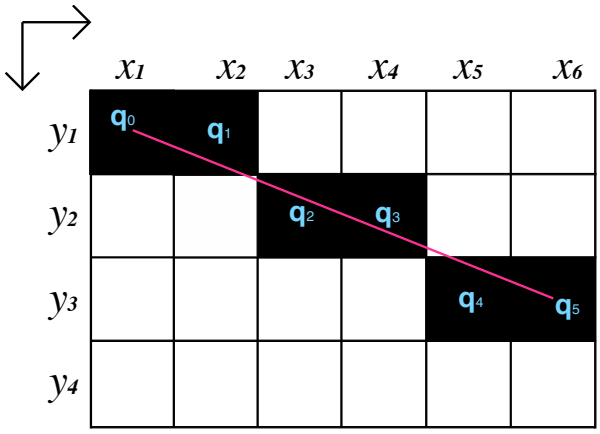


FIGURE 1.18 – *Représentation d'un segment discret dans l'octant 1.*

- $d(\boldsymbol{p},\boldsymbol{q}) \geqslant 0$ *et* $d(\boldsymbol{p},\boldsymbol{q}) = 0 \Leftrightarrow \boldsymbol{p} = \boldsymbol{q}$ *(définie positive)*
- $d(\boldsymbol{p},\boldsymbol{q}) = d(\boldsymbol{q},\boldsymbol{p})$ *(symétrique)*
- $d(\boldsymbol{p},\boldsymbol{r}) \leqslant d(\boldsymbol{p},\boldsymbol{q}) + d(\boldsymbol{q},\boldsymbol{r})$ *(inégalité triangulaire)*

La mesure la plus intuitive est la distance euclidienne entre deux points discrets.

Définition 1.14 *La distance euclidienne entre* $\boldsymbol{p}$ *et* $\boldsymbol{q}$ *appartenant à* $\mathbb{Z}^2$ *est définie par :*

$$\begin{array}{rccl} d_E & : & \mathbb{Z}^2 \times \mathbb{Z}^2 & \longrightarrow & \mathbb{R}^+ \\ & & (\boldsymbol{p},\boldsymbol{q}) & \longmapsto & \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2} \end{array}$$

Par la suite, nous utiliserons davantage la distance euclidienne au carré d_E^2 car elle présente l'avantage de ne considérer que des quantités entières.

Définition 1.15 *La distance euclidienne au carré entre* $\boldsymbol{p}$ *et* $\boldsymbol{q}$ *appartenant à* $\mathbb{Z}^2$ *est définie par :*

$$\begin{array}{rccl} d_E^2 & : & \mathbb{Z}^2 \times \mathbb{Z}^2 & \longrightarrow & \mathbb{N} \\ & & (\boldsymbol{p},\boldsymbol{q}) & \longmapsto & (x_q - x_p)^2 + (y_q - y_p)^2 \end{array}$$

Ces différentes distances sont à l'origine du calcul de transformées en distance à partir d'une image. Soit une distance d et une image binaire $\mathcal{I}$ contenant une forme $\mathcal{F}$.

Définition 1.16 *Une transformée en distance, notée DT, est une copie de* $\mathcal{I}$ *dans laquelle chaque pixel* $\boldsymbol{p}$ *de* $\mathcal{F}$ *est associé à sa distance au fond* $\bar{\mathcal{F}}$ *grâce à la formule suivante :*

$$\hat{d}_E(\boldsymbol{p},\bar{\mathcal{F}}) = \min\{d(\boldsymbol{p},\boldsymbol{q}) : \boldsymbol{q} \in \bar{\mathcal{F}}\}$$

Certains travaux ont pour but d'approximer la distance euclidienne dans le domaine discret. C'est le cas de Danielsson (1980) qui s'appuie sur la distance de chanfrein, ou de Ragnemalm (1993) qui utilise des mises à jour locales. Le problème principal de ces méthodes est qu'elles produisent une distance approchée, donc inexacte alors que la distance euclidienne au carré est une distance exacte. C'est la raison pour laquelle nous utilisons la distance euclidienne au carré comme distance pour construire une transformée en distance.

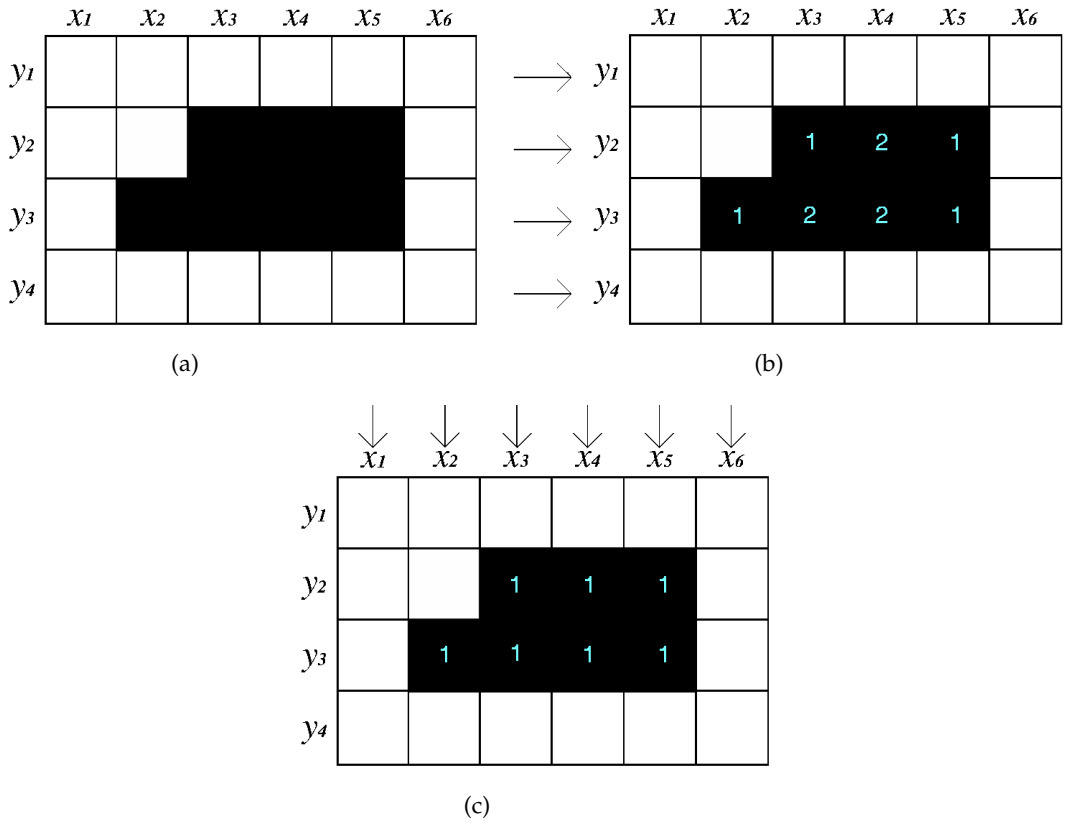


FIGURE 1.19 – *Illustration de la SEDT obtenue à partir de Meijster et al. (2002) et de Coeurjolly et Montanvert (2007), (a) la forme, (b) la carte G résultant de la première étape, (c) la transformée en distance euclidienne au carré finale.*

1.5.1 Transformée en Distance Euclidienne au Carré ($SEDT$)

La $SEDT$ (pour *Squared Euclidean Distance Transform*) de la forme $\mathcal{F}$ est calculable à partir de la méthode de Meijster et al. (2002), reprise par Coeurjolly et Montanvert (2007).

Cette méthode est basée sur un algorithme séparable (*i.e.* il procède par passes successives sur chaque dimension). Il a donc une complexité linéaire en n, le nombre de pixels de $\mathcal{I}$. C'est spécifiquement pour ces raisons que nous avons choisi d'utiliser cette méthode dans le cadre de la thèse.

Comme indiqué précédemment, ce calcul peut être effectué en deux étapes :

– La première étape permet de construire la carte G, qui est une carte temporaire. Elle est obtenue à l'aide d'une passe horizontale (sur les lignes). Plus précisément, chaque ligne est parcourue de gauche à droite, puis de droite à gauche pour déterminer la distance euclidienne de chacun des pixels de la forme au pixel de bord le plus proche sur la ligne (Algorithme 3).

  Plus formellement :

$$g(x_p, y_p) = \min_x \{|x_p - x| : 0 \leqslant x < M \text{ et } (x, y_p) \in \bar{\mathcal{F}}\}$$

Un exemple obtenu après cette étape est visible sur la Figure 1.19(b).

Algorithme 3: Pseudo-code de la transformée optimale en distance euclidienne : étape 1 (suivant l'axe x)

Entrées : Image binaire $\mathcal{I}$ de taille $M \times N$ et forme $\mathcal{F}$
Sorties : Carte G contenant la transformation en distance euclidienne suivant l'axe x

```
début
    pour j allant de 0 à N − 1 faire
        si (0, j) ∈ F̄ alors
            G(0, j) ← 0;
        sinon
            G(0, j) ← ∞;
        fin
        pour i allant de 1 à M − 1 faire
            si (i, j) ∈ F̄ alors
                G(i, j) ← 0;
            sinon
                /*G(i,j) est la distance au point de fond de gauche le plus proche*/
                G(i, j) ← 1 + G(i − 1, j);
            fin
        fin
        pour i allant de M − 2 à 0 faire
            si G(i + 1, j) < G(i, j) alors
                /*G(i,j) est la distance au point de fond de droite le plus proche*/
                G(i, j) ← 1 + G(i + 1, j);
            fin
        fin
    fin
fin
```

– Le but de la seconde étape est de construire la SEDT finale. En partant de la carte G, deux passes sont effectuées sur les colonnes (une du haut vers le bas et une autre du bas vers le haut). L'idée générale est de calculer pour chaque colonne x_i, toutes les paraboles $\mathcal{A}_y^{x_i}$ avec :

$$\mathcal{A}_y^{x_i}(y_i) = g(x_i, y)^2 + (y_i - y)^2$$

où $0 \leqslant y < N$ et $g(x_i, y)$ est obtenu à partir de l'étape 1.

Ces paraboles contiennent toutes les distances possibles aux points de fond les plus proches. Pour chaque colonne x_i, nous allons calculer M paraboles. Autrement dit, nous allons calculer autant de paraboles qu'il y a de pixels dans l'image.

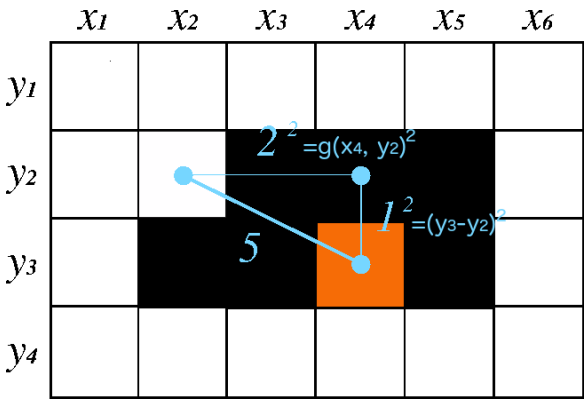


FIGURE 1.20 – *Représentation de la valeur associée à* $\mathcal{A}_{y_2}^{x_4}(y_3)$.

Prenons l'exemple de la parabole $\mathcal{A}_{y_2}^{x_4}$ de la Figure 1.21. Cette parabole donne des valeurs de distance euclidienne au carré au fond pour chacun des pixels de la colonne x_4, en tenant compte de la plus proche distance au fond dans la dimension horizontale du point (x_4, y_2).
Voici les valeurs obtenues pour la parabole $\mathcal{A}_{y_2}^{x_4}$ en fonction de y :

$$\mathcal{A}_{y_2}^{x_4}(y_1) = g(x_4, y_2)^2 + (y_1 - y_2)^2 = 5$$
$$\mathcal{A}_{y_2}^{x_4}(y_2) = g(x_4, y_2)^2 + (y_2 - y_2)^2 = 4$$
$$\mathcal{A}_{y_2}^{x_4}(y_3) = g(x_4, y_2)^2 + (y_3 - y_2)^2 = 5$$
$$\mathcal{A}_{y_2}^{x_4}(y_4) = g(x_4, y_2)^2 + (y_4 - y_2)^2 = 8$$

$\mathcal{A}_{y_2}^{x_4}(y_3)$ a donc comme valeur la distance euclidienne au carré entre (x_2, y_2) et (x_4, y_3), comme l'illustre la figure 1.20.
Pour n'obtenir, pour chaque pixel de l'image, que la distance au bord le plus proche, nous conservons, pour chaque colonne x_p, uniquement les valeurs des points associés aux y se trouvant sur l'enveloppe inférieure des paraboles $\mathcal{A}_y^{x_p}$, pour $0 \leqslant y < M$ (en rouge sur la Figure 1.21.

Plus formellement :

$$sedt(x_p, y_p) = \min\{g(x_p, y)^2 + (y_p - y)^2 : 0 \leqslant y < N\}$$

L'algorithme 4 est l'algorithme utilisé pour cette seconde étape.
Dans cet Algorithme, *sommetParabole* et *intersectParabole* sont des tableaux. Le tableau *sommetParabole* permet d'avoir en mémoire l'index des lignes des sommets des paraboles qui appartiennent à l'enveloppe inférieure. *intersectParabole* permet de stocker l'index des lignes de l'intersection des paraboles dont la référence à leur sommet se trouve dans *sommetParabole*. Les lignes 5 à 9 permettent de vérifier que la parabole en mémoire (celle dont le sommet est à la ligne $sommetParabole[z]$), est bien en-dessous de la parabole testée (celle dont le sommet est à la ligne j). Si cela n'est pas le cas, cela signifie que la parabole en mémoire ne fait plus partie de l'enveloppe inférieure. Par contre, la parabole testée est candidate pour faire partie de cette enveloppe (lignes 10 à 17). La

Algorithme 4: Pseudo-code de la transformation optimale en distance euclidienne au carré : étape 2 en suivant l'axe y.

Entrées : Carte G contenant la transformation en distance euclidienne suivant l'axe x.
Sorties : $SEDT$ représentant la Transformée en Distance euclidienne au Carré suivant les deux dimensions

début
  pour tous les $i \in [0..M-1]$ **faire**
    $z \leftarrow 0$; $sommetParabole[0] \leftarrow 0$; $intersectParabole[0] \leftarrow 0$;
    pour j *allant de* 1 *à* $N-1$ **faire**
      tant que $(z \geqslant 0)$ *et* $(\mathcal{A}^i_{sommetParabole[z]}(intersectParabole[z]) > \mathcal{A}^i_j(intersectParabole[z]))$ **faire**
        $z \leftarrow z-1$;
      fin
      si $z < 0$ **alors**
        $z \leftarrow 0$; $sommetParabole[0] \leftarrow j$;
      sinon
        $w \leftarrow 1 + Sep^i(sommetParabole[z], j)$;
        si $w < N$ **alors**
          $z \leftarrow z+1$;
          $sommetParabole[z] \leftarrow j$;
          $intersectParabole[z] \leftarrow w$;
        fin
      fin
    fin
    pour j *allant de* $N-1$ *à* 0 **faire**
      $SEDT(i,j) \leftarrow \mathcal{A}^i_{sommetParabole[z]}(j)$;
      si $j < intersectParabole[z]$ **alors**
        $z \leftarrow z-1$;
      fin
    fin
  fin
fin

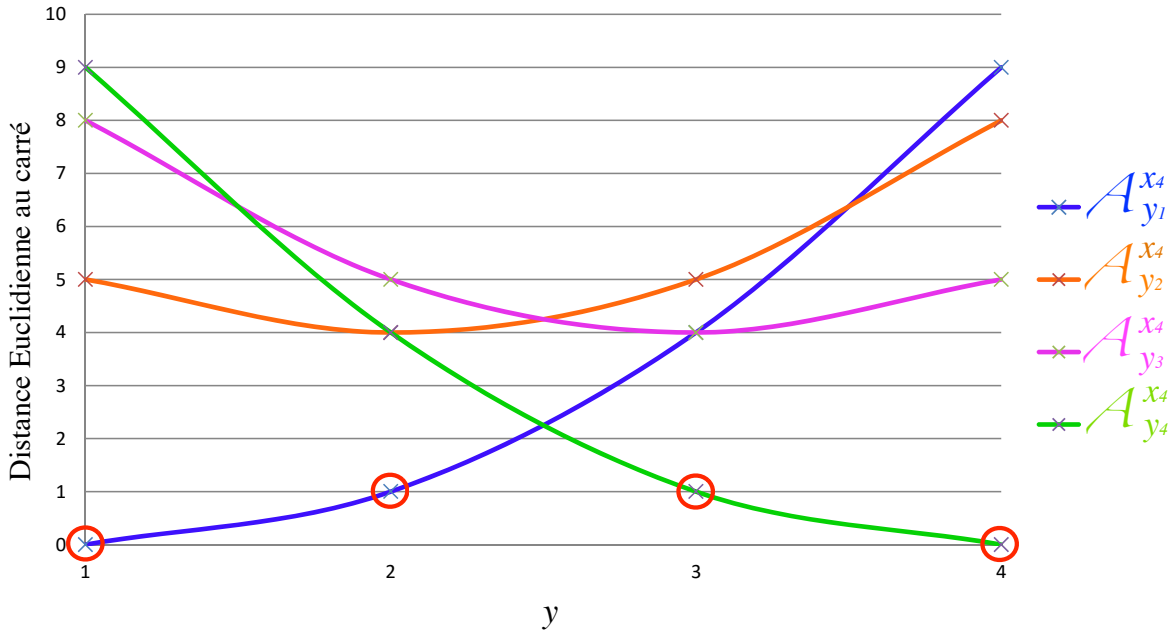


FIGURE 1.21 – *Illustration de l'étape 2 concernant la quatrième colonne. Les points appartenant à l'enveloppe inférieure sont entourés en rouge.*

ligne au niveau de laquelle elle s'intersecte avec la parabole acceptée précédemment est stockée dans w (ligne 11). Sur cette ligne, $Sep^i(u,v) = (v^2 - u^2 + G(i,v)^2 - G(i,u)^2)\ div\ (2(v-u))$ où div est la division entière. Si l'intersection se produit dans l'image, la ligne correspondant au sommet de la parabole testée est ajoutée à *sommetParabole* et la ligne de l'intersection avec la parabole précédente est ajoutée à *intersectParabole* (lignes 12 à 16). Dans les lignes 20 à 24, les paraboles enregistrées dans l'enveloppe inférieure sont dépilées une par une et la carte $SEDT$ est complétée.

1.5.2 Transformée en Distance Euclidienne Séquentielle Signée

Une autre manière de représenter la distance exacte entre un pixel de la forme et un pixel du fond est la $8SSED$ (Distance Euclidienne Séquentielle Signée) publiée par Ye (1988). Dans la transformée en Distance Euclidienne Séquentielle Signée, un pixel de l'image est associé à un vecteur qui représente le déplacement pour aller au point de fond le plus proche. Il ne s'agit donc pas d'une transformée en distance à proprement parler.

$$8SSED(x_p, y_p) = (8SSED_x(x_p, y_p), 8SSED_y(x_p, y_p))$$

où $8SSED_x(x_p, y_p)$ et $8SSED_y(x_p, y_p)$ sont des entiers et représentent respectivement le déplacement sur l'axe des x et sur l'axe des y.

La distance réelle entre un pixel $\boldsymbol{p}$ et $\bar{\mathcal{F}}$ est représentée par la longueur du vecteur défini par :

$$\| 8SSED(x_p, y_p) \| = \sqrt{8SSED_x^2(x_p, y_p) + 8SSED_y^2(x_p, y_p)}$$

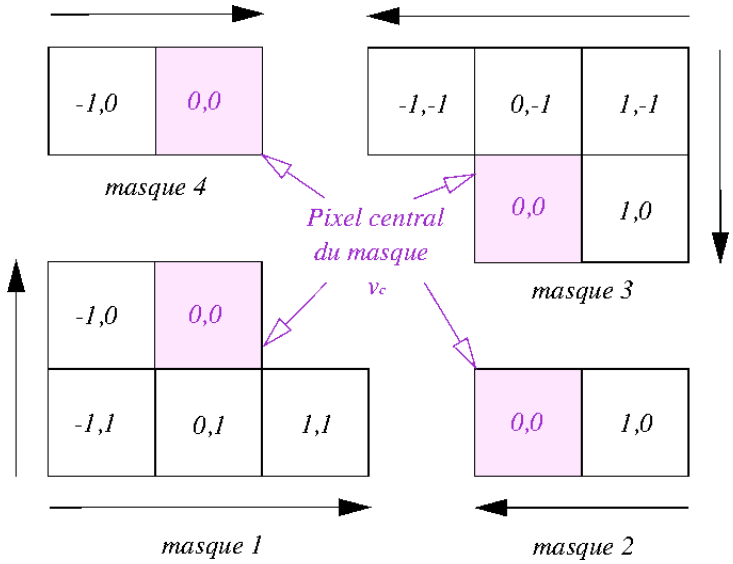


FIGURE 1.22 – *Masques utilisés dans la 8SSED.*

Pour un vecteur $(8SSED_x(x_p, y_p), 8SSED_y(x_p, y_p))$ en (x_p, y_p), les coordonnées du point $\boldsymbol{q}$ le plus proche au fond $\bar{\mathcal{F}}$ sont déterminées par :

$$(x_q, y_q) = ((x_p, y_p) + ((8SSED_x(x_p, y_p), 8SSED_y(x_p, y_p))$$

Tout d'abord, l'image est parcourue et la carte 8SSED est initialisée de la manière suivante (*cf.* Figure 1.23(a)) :

$$8SSED(x_p, y_p) = \begin{cases} (\infty, \infty) \text{ si } \boldsymbol{p} \in \mathcal{F} \\ (0, 0) \text{ sinon} \end{cases}$$

Puis, quatre masques (*cf.* Figure 1.22) sont déplacés dans la carte de distance initialisée en suivant l'Algorithme 5 pour la mettre à jour.

1.6 BOULES MAXIMALES DISCRÈTES ET AXE MÉDIAN DISCRET

LES boules maximales sont des objets mathématiques permettant de définir le squelette que nous allons en particulier étudier dans cette thèse.

Définition 1.17 (boule maximale discrète) *Soit $d_E^2 : \mathbb{Z}^2 \times \mathbb{Z}^2 \longrightarrow \mathbb{N}$ la distance euclidienne au carré discrète telle que définie dans la partie 1.5.*
Une boule de centre $\boldsymbol{c}$ et de rayon r relative à la distance d_E^2 est définie par :

$$B^<(\boldsymbol{c}, r) = \{\boldsymbol{q} \in \mathbb{Z}^2 | d_E^2(\boldsymbol{c}, \boldsymbol{q}) < r^2\}$$

Une boule maximale discrète est une boule discrète (disque) contenue dans la forme, non entièrement recouverte par une autre boule discrète de rayon supérieur, elle aussi contenue dans la forme.

La Figure 1.24(a) illustre ce concept.

Algorithme 5: Obtention de la transformée en distance 8SSED.

Entrées : Image $\mathcal{I}$ de taille $M \times N$ et forme $\mathcal{F}$, **Sortie :** Transformée en distance 8SSED
Hypothèse : Aucun pixel de $\mathcal{F}$ n'appartient au bord de l'image.
*/*Initialisation de la carte 8SSED*/*
pour tous les *les pixels* $\boldsymbol{p}$ *de* $\mathcal{I}$ **faire**
  si $\boldsymbol{p} \in \mathcal{F}$ **alors**
    $8SSED(x_p, y_p) \leftarrow (\infty, \infty)$;
  sinon
    $8SSED(x_p, y_p) \leftarrow (0, 0)$;
  fin
fin
pour *j allant de N-2 à 1* **faire**
  */*Application du masque 1 de gauche à droite*/*
  pour *i allant de 1 à M-2* **faire**
    pour tous les *les pixels* $\boldsymbol{q} \in \{(i-1, j), (i-1, j+1), (i, j+1), (i+1, j+1)\}$ **faire**
      si $\| 8SSED(x_q, y_q) \| > \sqrt{(8SSED_x(x_q, y_q) + (x_q - i))^2 + (8SSED_y(x_q, y_q) + (y_q - j))^2}$ **alors**
        $8SSED_x(x_q, y_q) = 8SSED_x(x_q, y_q) + (x_q - i)$;
        $8SSED_y(x_q, y_q) = 8SSED_y(x_q, y_q) + (y_q - j)$;
      fin
    fin
  fin
  */*Application du masque 2 de droite à gauche*/*
  pour *i allant de M-2 à 1* **faire**
    $\boldsymbol{q} \leftarrow (i+1, j)$;
    si $\| 8SSED(x_q, y_q) \| > \sqrt{(8SSED_x(x_q, y_q) + (x_q - i))^2 + (8SSED_y(x_q, y_q) + (y_q - j))^2}$ **alors**
      $8SSED_x(x_q, y_q) = 8SSED_x(x_q, y_q) + (x_q - i)$;
      $8SSED_y(x_q, y_q) = 8SSED_y(x_q, y_q) + (y_q - j)$;
    fin
  fin
fin
pour *j allant de 1 à N-2* **faire**
  */*Application du masque 3 de droite à gauche*/*
  pour *i allant de M-2 à 1* **faire**
    pour tous les *les pixels* $\boldsymbol{q} \in \{(i-1, j-1), (i, j-1), (i+1, j-1), (i+1, j)\}$ **faire**
      si $\| 8SSED(x_q, y_q) \| > \sqrt{(8SSED_x(x_q, y_q) + (x_q - i))^2 + (8SSED_y(x_q, y_q) + (y_q - j))^2}$ **alors**
        $8SSED_x(x_q, y_q) = 8SSED_x(x_q, y_q) + (x_q - i)$;
        $8SSED_y(x_q, y_q) = 8SSED_y(x_q, y_q) + (y_q - j)$;
      fin
    fin
  fin
  */*Application du masque 4 de gauche à droite*/*
  pour *i allant de 1 à M-2* **faire**
    $\boldsymbol{q} \leftarrow (i-1, j)$;
    si $\| 8SSED(x_q, y_q) \| > \sqrt{(8SSED_x(x_q, y_q) + (x_q - i))^2 + (8SSED_y(x_q, y_q) + (y_q - j))^2}$ **alors**
      $8SSED_x(x_q, y_q) = 8SSED_x(x_q, y_q) + (x_q - i)$;
      $8SSED_y(x_q, y_q) = 8SSED_y(x_q, y_q) + (y_q - j)$;
    fin
  fin
fin
retourner $8SSED$

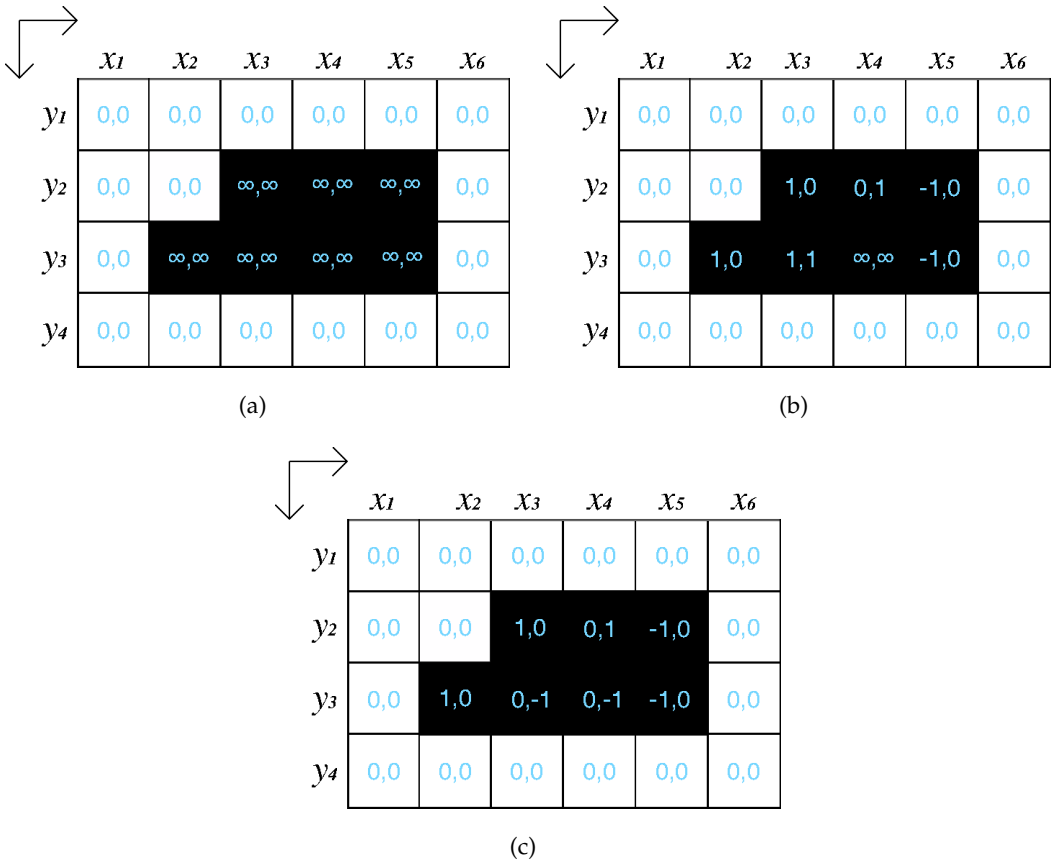


FIGURE 1.23 – *Illustration de la 8SSED obtenue à partir de Ye (1988), (a) initialisation, (b) étape 1 (déplacement des masques 1 et 2), (c) étape 2 (déplacement des masques 3 et 4).*

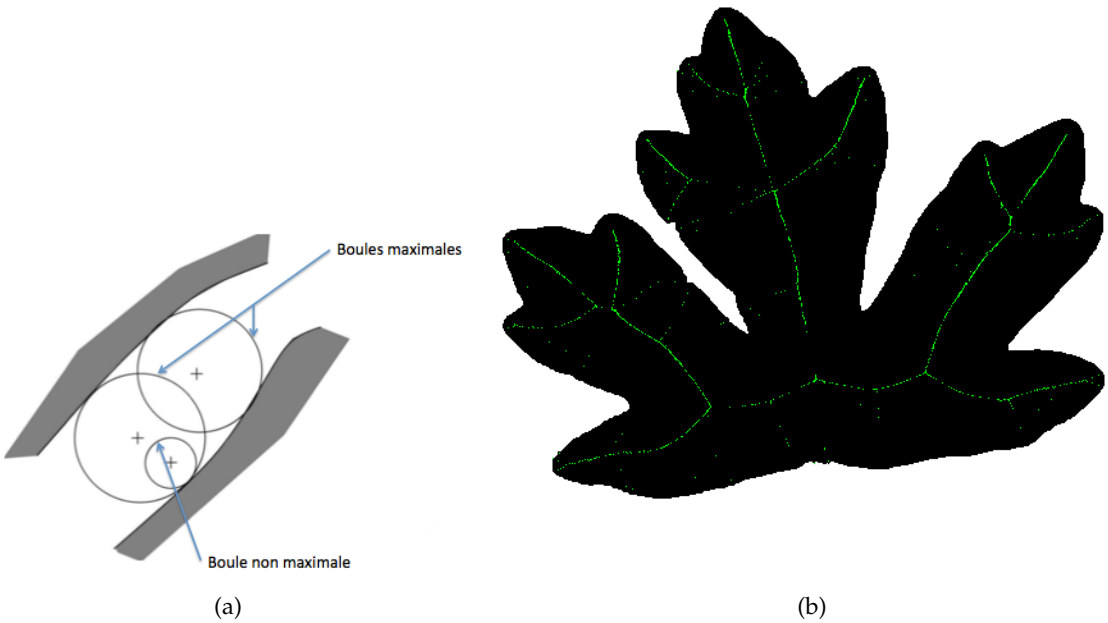


FIGURE 1.24 – *(a) Illustration du concept de boule maximale (Coeurjolly et Montanvert (2007)), (b) Axe médian d'une feuille.*

L'ensemble des centres des boules maximales discrètes est l'axe médian discret (*cf. Figure 1.24(b)*). Une construction de ce dernier a été proposé par Coeurjolly et Montanvert (2007), Coeurjolly et al. (2007). Nous la décrirons plus précisément dans le Chapitre 3. À noter que l'axe médian discret n'est pas homotope à la forme car il est impossible de passer de l'un à l'autre par une déformation continue, mais en revanche, il est un codage réversible. En d'autres termes, la forme peut être reconstruite à partir de l'ensemble des centres des boules maximales. L'intérêt de ce descripteur réside dans le fait que l'axe médian auquel ont été associées des valeurs de distance est une compression de la forme. Pour la décompresser, il suffit de la reconstruire.

Notons $B(\boldsymbol{p}, r)$ l'ensemble des points de $\mathbb{Z}^2$ qui peuvent être reconstruits à partir de la boule (maximale) B centrée en $\boldsymbol{p}$ et de rayon r :

$$B(\boldsymbol{p}, r) = \{\boldsymbol{q} \in \mathcal{F} | d(\boldsymbol{p}, \boldsymbol{q}) \leq r\}$$

où d est la distance utilisée dans la carte de distance.

Conclusion

Les bases de la géométrie discrète étant posées, l'espace discret dans lequel nous allons travailler est suffisamment bien connu pour aborder la suite de cette thèse. En effet, le pavage d'une image numérique a été décrit, ainsi que l'organisation des pixels au sein d'une image binaire. Ceci nous a permis d'expliciter la notion de forme. Puis, nous avons abordé la topologie pour expliquer une notion très utilisée en squelettisation, à savoir le point simple. Une forme peut être décrite par son contour, c'est la raison pour laquelle nous nous sommes quelque peu attardés sur la manière d'obtenir ce contour. Finalement, nous avons survolé le domaine des distances discrètes exactes, elles aussi très utilisées en squelettisation, notamment pour calculer une carte de distance. Nous avons également défini le terme d'axe médian, à partir des boules maximales, permettant de caractériser la forme et de la reconstruire.

Dans les chapitres suivants, nous ferons référence aux notions présentées dans cette partie.

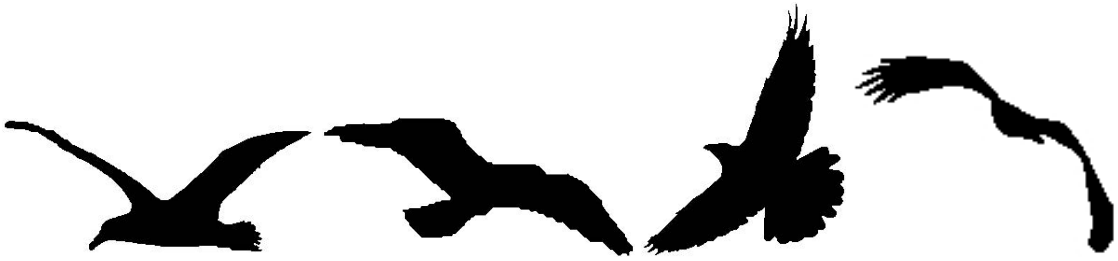
2 État de l'art de la description à l'appariement des formes planes

Sommaire

2.1 Squelettes 37
2.1.1 Méthodes basées sur un amincissement topologique 38
2.1.2 Méthodes basées sur la carte de distance 40
2.1.3 Méthodes basées sur le diagramme de Voronoï 42
2.2 Hiérarchisation et élagage du squelette 46
2.2.1 Hiérarchisation par relation de parenté 47
2.2.2 Hiérarchisation par approche multirésolution 51
2.2.3 Hiérarchisation sous-entendue 52
2.2.4 Une conséquence : l'Élagage 54
2.3 Appariement du squelette 55
2.3.1 Graphe acyclique orienté (DAG pour *Directed Acyclic Graph*) 56
2.3.2 Graphe Relationnel Attribué (ARG) 58
2.3.3 Appariement par la construction d'un arbre itérativement 60
2.3.4 Appariement tenant compte du contexte 62
Conclusion 62

L'analyse de formes planes nécessite l'extraction de descripteurs. Ils sont le résultat de transformations mathématiques du contour et/ou de l'intérieur de la forme. Un descripteur de forme doit caractériser la forme au mieux et permettre de quantifier facilement la similarité entre deux formes. Pour ce faire, il doit être compact, robuste au bruit et aux transformations affines simples telles que la rotation, la mise à l'échelle, la translation. Une des difficultés est que nous souhaitons reconnaître des objets réels à partir de leur projection 2D sur le plan image. Prenons l'exemple d'un oiseau en train de voler, la forme plane varie en fonction du point de vue à partir duquel est capturé l'objet (Figure 2.1). Un bon descripteur de forme doit donc tolérer les différences géométriques des objets d'une même catégorie, les occultations, les distorsions mais à la fois permettre de discriminer les objets de différentes classes. En résumé, dans l'idéal, un descripteur de forme devrait donc avoir les caractéristiques suivantes :

- discrimination ;
- informativité ;
- invariance aux transformations géométriques (translation, changement d'échelle, rotation, symétrie...) ;
- faible sensibilité aux déformations non-linéaires (bruit, articulation, occultation) ;
- compacité ;
- facilité d'extraction et d'utilisation ;
- capacité à distinguer l'importance des différentes parties dans la forme.



(a) (b) (c) (d)

FIGURE 2.1 – *Illustration de formes variant considérablement en fonction des déformations (Yang et al. (2016)).*

Nous avons choisi de ne pas décrire exhaustivement les divers descripteurs existants mais plutôt de les classer pour obtenir une vue globale de l'existant et de positionner le squelette, descripteur que nous avons choisi dans cette thèse, dans ce domaine.

Classiquement, les descripteurs peuvent être catalogués en deux classes, chacune étant divisée en sous-classes (Zhang et Lu 2004).

1. **Les descripteurs basés sur le contour** : ces derniers exploitent seulement les informations du contour de la forme. L'objet est alors vu comme une collection de tous ses points situés à la frontière avec le fond de l'image. Parmi ceux-ci, il existe plusieurs types d'approches permettant de les calculer.
   (a) **Les méthodes locales** : dans ces techniques, le bord est divisé en segments (appelés *primitives*). Les méthodes diffèrent dans la sélection des primitives et dans leur

organisation pour la représentation des formes. La représentation est souvent une chaîne de caractères ou un graphe. Par exemple,

- **la "Chain Code"** : la forme est décrite par une séquence de segments de ligne unitaires munis d'une orientation (Freeman 1961) ;
- **la décomposition polygonale** : le contour est sectionné en segments de ligne grâce à une approximation polygonale (Groskey et Mehrotra 1990, Groskey et al. 1992, Debled-Rennesson 1995) ;
- **la décomposition selon la courbure** : le contour est sectionné en utilisant les points au niveau desquels la courbure s'inverse (Berretti et al. 2000).

(b) **Les méthodes globales** : dans ces techniques, la forme est vue comme un vecteur de propriétés dérivé de l'intégralité du contour. Par exemple,

- **les descripteurs de formes basiques**, tels que le périmètre, la moyenne de l'énergie de flexion, le ratio de circularité, le centre de gravité, l'axe de moindre inertie, l'excentricité, la variance d'ellipse (Young et al. 1974, Peura et Iivarinen 1997). Il s'agit souvent de propriétés géométriques qui ne discriminent que les formes très différentes. Ils peuvent justement être utilisés lors d'une étape de prétraitement pour éliminer les formes trop différentes ;
- **les moments frontière** : il s'agit d'intégrales calculées le long des bords représentés par des segments (Chen 1993).

(c) **Les méthodes "riches"** : Par exemple, on trouve dans la littérature

- **les contexte de formes (*shape context*)** : il s'agit d'un descripteur local dans lequel sont incorporées des informations globales. Le principe de l'algorithme est d'échantillonner le contour de la forme, et d'obtenir pour chacun de ces points le contexte de forme en déterminant la distribution relative des points les plus proches grâce à un histogramme de distribution de coordonnées log-polaires (Belongie et al. 2002, Ling et Jacobs 2007, Xie et al. 2008). Cette technique est notamment performante lorsque les formes sont bruitées ou floutées (Chatbri et al. (2015)). Une extension utilisant des fonctions de hauteur est proposée par Wang et al. (2012). Le principe est de calculer une longueur signée entre chaque point échantillonné du bord et une droite ;
- **la représentation multi-échelle** : étude des courbures du bord de la forme à différentes échelles (Mokhtarian et Mackworth 1992, Mokhtarian 1995, Mokhtarian et al. 1997, Alajlan et al. 2007, Adamek et O'Connor 2004).

2. **Les descripteurs basés sur la région** : ces techniques tiennent compte de tous les pixels à l'intérieur de la forme dans l'obtention de la représentation. Tout comme les méthodes basées sur le contour, elles peuvent être divisées en sous-classes.

(a) **Les méthodes locales** : dans ces techniques, la forme est divisée en parties.

- **l'enveloppe convexe** : il s'agit de la plus petite région convexe contenant la forme (Davies 2004, Sonka et al. 2014) ;
- **le squelette** : ensemble des pixels médians connectés de la forme. Ce descripteur fait l'objet de la Partie 2.1.

(b) **Les méthodes globales** : dans ces techniques, la forme est traitée comme un tout et la représentation en résultant est souvent un vecteur d'attributs numériques. Par exemple,

- **les moments région :** mesure quantitative spécifique basée sur les intégrales calculées sur la forme regroupant les moments invariants (Zhang et al. 2002), les moments de Zernike (Teh et Chin 1988), les moments algébriques (Taubin et Cooper 1991), *etc.*
- **le descripteur de Fourier générique** : il est acquis en appliquant une transformée de Fourier sur une image à coordonnées polaires échantillonnée (Zhang et Lu 2002) ;
- **les matrices de formes** : "grilles" rectangulaires, circulaires, *etc.* permettant la représentation de formes échantillonnées (Goshtasby 1985) ;
- **les descripteurs topologiques** : techniques s'intéressant aux attributs d'une forme invariants à des mouvements d'étirement ou de rétrécissement, ne devant pas causer de coupure de la forme ou de fusion de deux composantes séparées au départ. Parmi ces descripteurs, nous pouvons citer le nombre d'Euler (Sossa-Azuela et al. 1996) ou le graphe de Reeb (Reeb 1946).

Des informations plus précises concernant l'ensemble des descripteurs de formes peuvent être retrouvées dans les articles de Bober (2001), Yang et al. (2008), Loncaric (1998), Zhang et Lu (2004).

Le squelette, ensemble de points médians connectés de la forme, que nous avons choisi d'étudier dans cette thèse, est un descripteur singulier. En effet, il se base sur les points du contour mais est situé à l'intérieur de la forme.

Par ailleurs, en général, les méthodes locales ont pour inconvénient d'être trop sensibles au bruit. En conséquence, ces méthodes peuvent ne pas être efficaces pour la reconnaissance de formes lorsque les formes sont complexes, ou comprennent un réarrangement des parties. Au contraire, les techniques globales, basées sur les informations relatives aux pixels situés à l'intérieur de la forme, capturent les caractéristiques globales de celle-ci. Elles sont donc robustes aux déformations locales mais ont du mal à capturer les détails locaux du bord de la forme. Leur pouvoir de discrimination est donc limité lorsqu'il y a beaucoup de variations intra-classes. Ainsi, comme l'expliquent Petrakis et al. (2002), les descripteurs de forme ayant seulement des informations globales ou locales ont plus de chances de ne pas être assez robustes face aux diverses situations dans la reconnaissance, d'autant plus qu'il est difficile de distinguer le bruit des détails locaux sur le bord de la forme ; d'où l'intérêt des méthodes mixtes.

Le squelette est un descripteur qui combine des informations topologiques et géométriques, ce qui le rend très intéressant lors de la déformation des formes, la présence d'articulations ou le réarrangement des parties (Sebastian et Kimia (2005)). Le squelette a, non seulement des propriétés élégantes, mais il induit aussi naturellement la forme. En effet, les résultats de Hung et al. (2012), publiés dans un journal de neurosciences, confirment les pré-

dictions théoriques de longue date, à savoir que le cerveau code les objets naturels grâce à une structure de squelette.

Ce descripteur étant au cœur de notre travail de thèse, une partie importante de cet état de l'art lui sera donc réservée (Partie 2.1). De nombreux travaux ont été consacrés à déterminer une hiérarchie entre les branches du squelette afin de maîtriser leur importance respective. Nous détaillerons donc les diverses manières employées pour arriver à ce résultat et verrons comment il est possible d'élaguer le squelette afin de faire disparaître les branches de moindre importance (Partie 2.2). Enfin, nous verrons les différentes utilisations du squelette lors de l'appariement de forme (Partie 2.3).

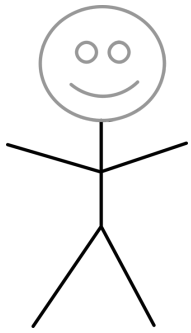
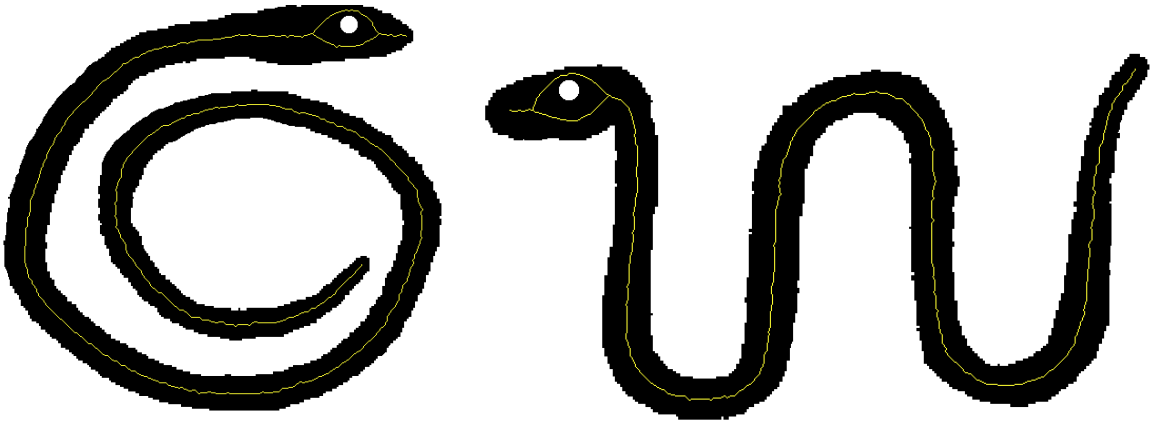


FIGURE 2.2 – *Le squelette est une représentation compacte intuitive du corps humain.*



(a) (b)

FIGURE 2.3 – *Représentation d'un serpent, associé à son squelette, suivant deux positions.*

2.1 SQUELETTES

LE squelette est un outil puissant pour obtenir un appariement de formes de qualité car il résume et aide à la compréhension de la forme et de sa topologie. La squelettisation d'une forme est un processus qui permet d'obtenir une représentation parlante, compacte, filiforme de la silhouette d'un objet et centrée dans celle-ci. Pour le mettre en évidence, prenons l'exemple de la représentation enfantine du corps humain (Figure 2.2). N'importe quelle personne regardant le dessin saura qu'il s'agit d'un bonhomme. Les parties qui nous intéressent pour appréhender le squelette sont le corps et les quatre membres représentés par des segments disposés en fonction de la morphologie humaine. Ces cinq segments forment le squelette d'un corps humain en faisant abstraction de la tête. Un exemple de squelette calculé sur la silhouette d'un serpent dans deux positions différentes est présenté sur la Figure 2.3. Les squelettes sont différents mais ont la même topologie.

Dans l'idéal, le squelette est un ensemble de points ayant les propriétés suivantes :

- homotope à la forme ;
- invariant par transformations affines (translation, rotation et changement d'échelle) ;
- centré dans la forme ;

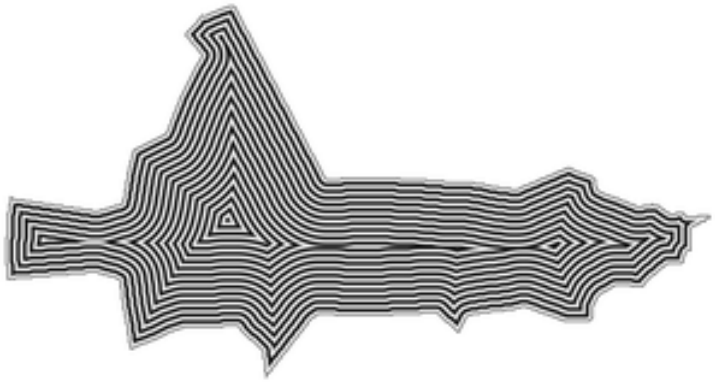


FIGURE 2.4 – *Illustration du feu de prairie.*

- robuste au bruit (absence de branches, provenant du bruit sur le bord de la forme et n'ayant pas d'intérêt (appelées *barbules*), se greffant sur le squelette proprement dit) ;
- ayant une épaisseur d'un pixel (mince ou fin) ;
- réversible (la reconstruction de la forme est possible) ;

Nous nous intéressons, dans la suite, aux différentes méthodes permettant de le calculer.

Le squelette a été introduit par Blum (1967) en utilisant le terme d'axe médian. Intuitivement, il s'agit de l'ensemble des points médians entre les bords, situés à l'intérieur de la forme. L'analogie avec le feu de prairie illustre le principe de construction de l'axe médian. Supposons que la forme soit une prairie et donc que les bords de la forme soient les bords de la prairie. Au commencement, supposons que tous les points du contour de la prairie soient simultanément en feu. Puis le feu se propage à l'intérieur du pré à vitesse constante dans la direction normale au contour. Le squelette est constitué des endroits où les fronts enflammés se rencontrent. Notons que l'instant de rencontre des fronts est lié à l'épaisseur de la forme. Ainsi, le front d'onde initialisé sur le contour et se propageant comporte des informations concernant le bord de la forme. De plus, comme il s'agit du déplacement d'un front d'ondes de manière uniforme, la connectivité initiale de la forme est préservée. En résumé, le squelette est une représentation filiforme de la forme qui préserve les propriétés topologiques et géométriques de celle-ci. La Figure 2.4 illustre le feu de prairie dans laquelle les fronts successifs sont dessinés alternativement en gris foncé et en gris clair pour être visibles.

Au fil du temps, les chercheurs s'intéressèrent de plus en plus au squelette de Blum (1967). Par exemple, Leymarie et Levine (1989; 1992) simulèrent le feu de prairie en utilisant un contour actif dit "snake" alors que Xia (1989) le modélisa de manière discrète, c'est-à-dire que chaque pixel "feu" se propage à ses voisins.

Ainsi, parallèlement, trois grandes catégories de méthodes de squelettisatison ont vu le jour.

2.1.1 Méthodes basées sur un amincissement topologique

Les méthodes de calcul de squelette ayant le plus de similitudes avec le feu de prairie sont celles basées sur un algorithme d'amincissement topologique pour générer des points médians. L'idée générale de cette catégorie de méthodes est de réduire l'aire de la forme gra-

duellement à partir du bord pour obtenir une forme fine (un pixel d'épaisseur) conservant la topologie de la forme. Autrement dit, il s'agit d'*éplucher* la forme petit à petit jusqu'à ce qu'on ne puisse plus enlever de pixels sans modifier la topologie. En termes plus scientifiques, l'amincissement topologique consiste à faire une érosion conditionnelle du contour de la forme jusqu'à obtenir une figure mince et centrée. Cette érosion ne doit ni modifier la topologie, ni retirer les extrémités des branches.

Les pixels supprimés doivent donc respecter deux conditions :

- La suppression du pixel ne modifie pas la topologie de la forme (des trous ne doivent pas apparaître et la forme ne doit pas être scindée). Le pixel doit donc être un point simple (*cf.* Chapitre 1).
- Le pixel n'est pas une extrémité de l'ensemble courant (dans le cas contraire, toute forme connexe serait réduite à un seul pixel), c'est-à-dire qu'il a plus d'un voisin appartenant à l'ensemble courant.

Ces pixels sont donc appelés points simples non-terminaux. Dans cette catégorie de méthodes, le traitement est purement local car pour vérifier ces deux conditions, il suffit d'examiner le voisinage du pixel traité. Il existe deux approches principales pour la suppression des pixels :

- **Approche séquentielle** : les points de la forme sont balayés séquentiellement jusqu'à ce qu'il n'y ait plus de suppression possible. Si un candidat est un point simple non-terminal alors il est supprimé tout de suite. Par conséquent, tous les autres tests de simplicité sur ses pixels voisins devront en tenir compte. Il est à noter que, dans cette approche, le squelette obtenu dépend du balayage de la forme puisque c'est lui qui donne l'ordre des candidats à la suppression. Donc, il se peut qu'au final, le squelette ne soit pas centré dans la forme. Néanmoins, la préservation de la topologie, du fait même de la définition d'un point simple, montre l'intérêt majeur de cette approche. Dans la littérature, de nombreux chercheurs ont travaillé sur cette approche, comme Palágyi (2014), Saeed et al. (2001), Saeed et al. (2010), *etc.*
- **Approche parallèle** : Cette approche est une érosion itérative couche par couche jusqu'à ce que l'amincissement soit stable. Lors d'une itération, les points de contour sont testés un à un. S'ils sont simples et non-terminaux, alors, ils sont marqués comme étant supprimables. Ils sont réellement supprimés quand tous les points de contour ont été testés. Ainsi, à chaque itération, certains points de contour sont supprimés simultanément. Un exemple d'amincissement topologique est présenté sur la Figure 3.8. Sur cette figure, trois passages ont été effectués. Lors du premier, les pixels rouges ont été supprimés. Lors du deuxième, ce sont les pixels verts et enfin, lors du troisième, les pixels bleus. Les pixels restants, qui forment le squelette, sont représentés en noir. L'intérêt d'utiliser cette approche est qu'elle fournit un squelette centré dans la forme. Cependant, la suppression en parallèle de points simples non-terminaux risque de ne pas préserver la topologie de la forme sauf en reformulant la notion de points simples. Certains auteurs se basent sur des motifs (Manzanera et al. (2002), Wu et Tsai (1992), Datta et Parui

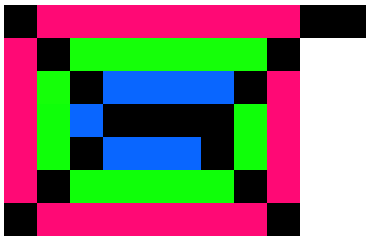


FIGURE 2.5 – *Exemple d'amincissement topologique parallèle.*

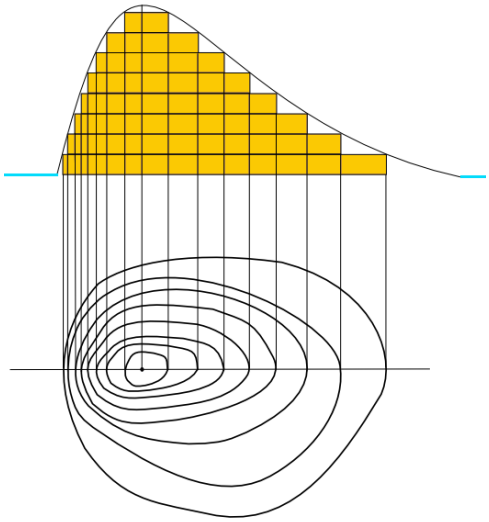


FIGURE 2.6 – *Courbes de niveaux et représentation discrète associée*

(1994)), règles (Rockett (2005), Ahmed et Ward (2002), Lohou et Dehos (2010)), ou noyaux critiques (Bertrand et Couprie 2014), *etc.*

Un état de l'art sur l'amincissement topologique a été proposé par Lam et al. (1992).

2.1.2 Méthodes basées sur la carte de distance

D'une manière générale, une carte de distance, également appelée "transformée en distance", peut être interprétée comme la carte de relief d'une île dans laquelle chaque pixel est étiqueté par l'altitude de l'île à cet endroit. Sur les cartes topographiques de l'IGN, tous les lieux situés à une même altitude sont représentés par des lignes de niveau. Ce concept est illustré par la Figure 2.6.

Dans notre cas, l'île représente la forme, et la hauteur en un point décrit sa distance au point de bord le plus proche. La Figure 2.7 montre une forme de reptile, la représentation en trois dimensions de sa carte de distance et son squelette associé. La carte de distance peut être générée par un front d'onde qui se propage à vitesse constante du bord de la forme vers l'intérieur. Les pixels atteints au même instant ont la même hauteur et forment une ligne de niveau. Cette dernière est caractérisée par des pentes convergeant vers des crêtes et des sommets. Nous appelons une crête, la ligne de points hauts d'un relief séparant deux versants opposés. Un sommet est un point culminant d'un relief (extremum local). Intuitivement, la détection

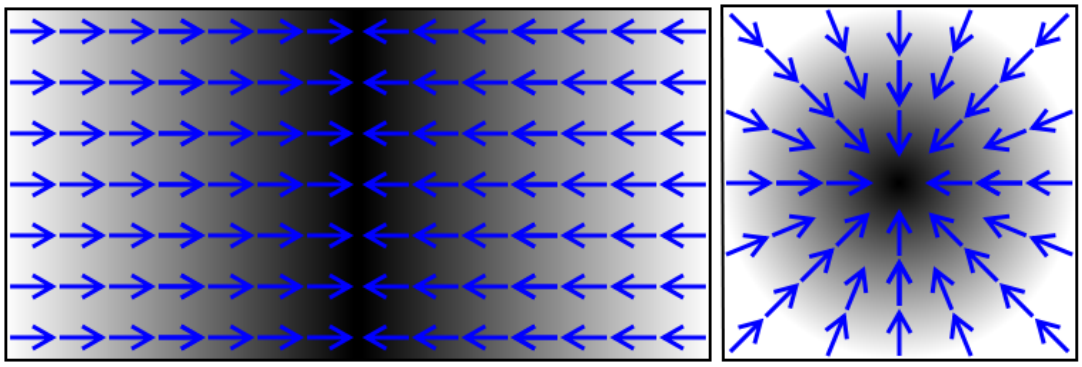
(a) (b) (c)

FIGURE 2.7 – *(a) Forme de reptile, (b) Représentation en trois dimensions de la carte de distance associée, (c) squelette obtenu à partir de la carte de distance.*

des crêtes et des sommets est une transformation du feu de prairie. Il s'agit d'un autre moyen d'extraire le squelette de la forme.

Ces algorithmes ont une démarche relativement similaire. Tout d'abord, la carte de distance est calculée. Il existe diverses cartes de distances permettant d'obtenir un squelette. Borgefors et Sanniti di Baja (1988) calculent une distance en nid d'abeille sur un plan hexagonal. Arcelli et Sanniti di Baja (1989) utilisent la distance d_4 représentant la longueur du plus petit chemin 4-connexe entre deux points. Quelques années plus tard, Thiel (1992) puis Attali et Thiel (1993) utilisent la distance d_8 (longueur du plus petit chemin 8-connexe entre deux points) et les distances de chanfrein calculées *via* des masques pondérés (Montanari (1968), Borgefors (1986)). De leur côté, Choi et al. (2003) emploient la Distance Euclidienne Signée de Ye (1988) expliquée dans le Chapitre 1. Dans leurs travaux, Coeurjolly et Montanvert (2007) se servent de la Carte de Distance Euclidienne au Carré (Meijster et al. (2002), exposée dans le Chapitre 1). Gorelick et al. (2006) créent une carte de distance *via* une "marche aléatoire". En d'autres termes, ils assignent à chaque point de la forme le temps requis pour sortir de la forme en utilisant des déplacements aléatoires (formalisés par l'Équation de Poisson). Cela permet d'obtenir une carte de distance moins sensible au bruit.

La deuxième étape consiste à déterminer les crêtes et les sommets de la carte de distance. Pour ce faire, Chang (2007), Malandain et Fernández-Vidal (1998), Siddiqi et al. (2002), Lebourgeois et Emptoz (2007) utilisent le gradient. Schématiquement, sur la représentation de la carte de distance en trois dimensions, lorsqu'il y a une crête (respectivement un sommet), la carte de distance forme un toit (respectivement un cône). Le gradient est une grandeur vectorielle spécifiant la variation d'une grandeur physique dans l'espace. Le gradient d'une carte de distance caractérise donc la pente en chaque point de la forme. Il est perpendiculaire à la ligne de niveau passant par le point duquel il est issu et se dirige vers le haut de la pente. Les points de crête et sommets sont des points particuliers pour lesquels le gradient n'est pas défini car la carte de distance euclidienne n'est pas dérivable sur les crêtes. La Figure 2.8(a) (respective-



(a) (b)

FIGURE 2.8 – *Gradient autour (a) d'une crête, (b) d'un sommet. La distance est représentée en niveaux de gris.*

ment la Figure 2.8(b)) représente le gradient d'une partie de la carte de distance ayant la forme d'un toit (respectivement d'un cône) lorsqu'elle est représentée en trois dimensions.

Arcelli et Sanniti di Baja (1992), Niblack et al. (1992), Shih et Pu (1995) se basent sur les valeurs de distances dans le voisinage pour détecter les crêtes et sommets. Quant à Arcelli et Sanniti di Baja (1993), Coeurjolly et Montanvert (2007), ils préfèrent se servir du centre des boules maximales qui permettent de déterminer les maxima locaux de la carte de distance (*cf.* Chapitre 1). Sanniti di Baja et Thiel (1996) ont une approche un petit peu plus hybride dans le sens où ils détectent les pics de la carte de distance grâce aux boules maximales et les points selle (minimum local d'une crête) grâce à la configuration locale. D'autres chercheurs ont des approches plus originales. Par exemple, Latecki et al. (2007a) proposent une approche s'appuyant sur la diffusion isotrope calculée sur un champ de gradients. Le squelette est obtenu en reliant les points critiques avec des chemins géodésiques. Choi et al. (2003) ont mis au point un critère de connexité basé sur les boules maximales et la configuration des voisins de chaque point. Kimmel et al. (1995) tiennent compte des courbures maximales du bord de la forme pour calculer leur squelette à partir de la carte de distance.

Lorsque la méthode génère des points de squelettes non connectés alors qu'ils devraient l'être, ceux-ci sont reliés grâce au gradient de la carte de distance dans la plupart des cas (Arcelli et Sanniti di Baja (1992), Latecki et al. (2007a), Sanniti di Baja et Thiel (1996), Arcelli et Sanniti di Baja (1993)). Lors de cette étape, de faux trous de quelques pixels peuvent apparaître, ce qui compromet l'homotopie à la forme. Ils sont alors identifiés pour procéder à une phase de remplissage.

Dans la plupart des cas, le squelette obtenu n'est pas fin, c'est-à-dire que son épaisseur est supérieure à un pixel. La dernière étape consiste donc à l'amincir. Une éventuelle ébarbulation (ou élagage) peut ensuite être envisagée pour supprimer les parties du squelette provenant d'irrégularités sur le contour de la forme.

2.1.3 Méthodes basées sur le diagramme de Voronoï

Cette troisième catégorie de méthodes calcule les axes de symétrie de la forme. Cette structure géométrique continue a été introduite par le mathématicien allemand Johann Peter Gus-

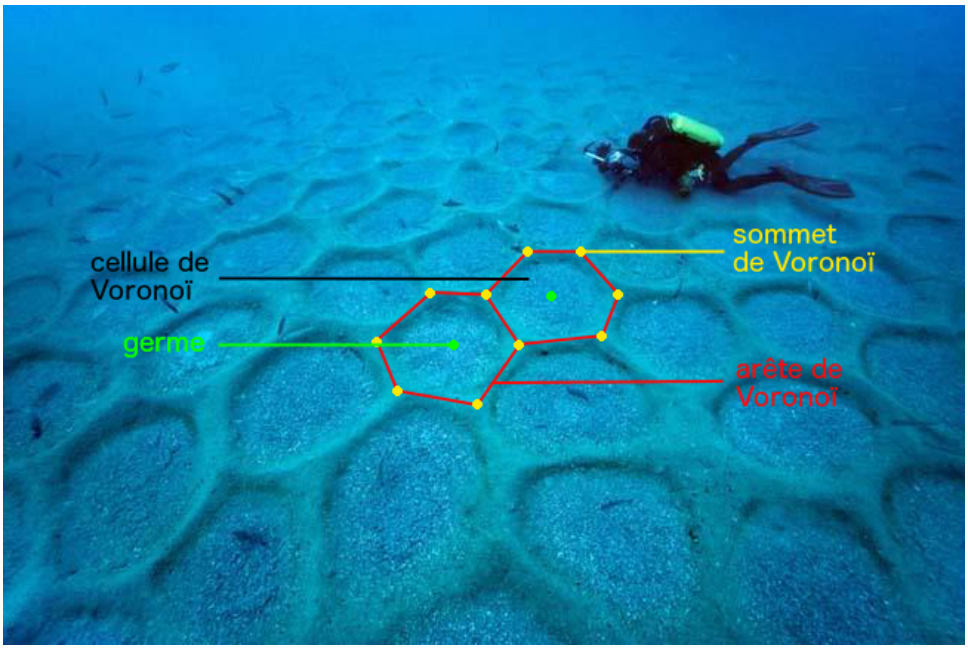


FIGURE 2.9 – *Création d'un diagramme de Voronoï lors de la reproduction du Picarel.*

tav Dirichlet en deux et trois dimensions. Il faudra attendre 1908 pour que le mathématicien russo-ukrainien Georgy Voronoï formalise cette notion dans le cas général.

Pour introduire ce concept, intéressons-nous au Picarel, un poisson marin des eaux côtières. Pour se reproduire, un grand nombre d'individus se rassemble sur le sable en bordure d'herbiers. Chaque mâle s'octroie un territoire en repoussant le sable à partir de l'endroit où il se trouve avec la même force que son voisin pour créer un nid. Un nid peut donc être considéré comme la "zone d'influence" du mâle qui l'a créée. En analysant le comportement de ces poissons, nous nous apercevons qu'ils partitionnent le fond marin pour avoir chacun leur territoire. En effet, un poisson mâle crée son nid pour que toute la surface soit plus proche de son lieu de ponte que des autres. Les poissons créent ainsi un diagramme de Voronoï dans lequel le nid est appelé cellule de Voronoï, le lieu de ponte représente le germe, le sable jouant le rôle de barrière entre deux nids constitue une arête de Voronoï (Figure 2.9).

Formellement, le diagramme de Voronoï est une partition de l'image $\mathcal{I}$.

Considérons un ensemble $\mathcal{G}$ de points dans $\mathcal{I}$ appelés germes. La cellule de Voronoï du point $\boldsymbol{p}_i \in \mathcal{G}$ correspond à l'ensemble de points de $\mathcal{I}$ plus proches de $\boldsymbol{p}_i$ que de tout autre germe. Le diagramme de Voronoï de $\mathcal{I}$ est l'ensemble des cellules dont les germes appartiennent à $\mathcal{I}$.

Le principe pour construire un diagramme de Voronoï est le suivant. Un front est propagé à partir de chaque germe à la même vitesse, en définissant une région circulaire. Quand deux fronts se rencontrent, une frontière linéaire se forme le long de la médiatrice entre deux germes associés (arête de Voronoï). Un sommet de Voronoï est l'intersection d'au moins trois cellules de Voronoï. Lorsque la croissance des régions n'est plus possible, une partition du plan est obtenue. Un exemple de diagramme de Voronoï est présenté sur la Figure 2.10.

Pour obtenir le squelette d'une forme, le bord de la forme est échantillonné et chaque point

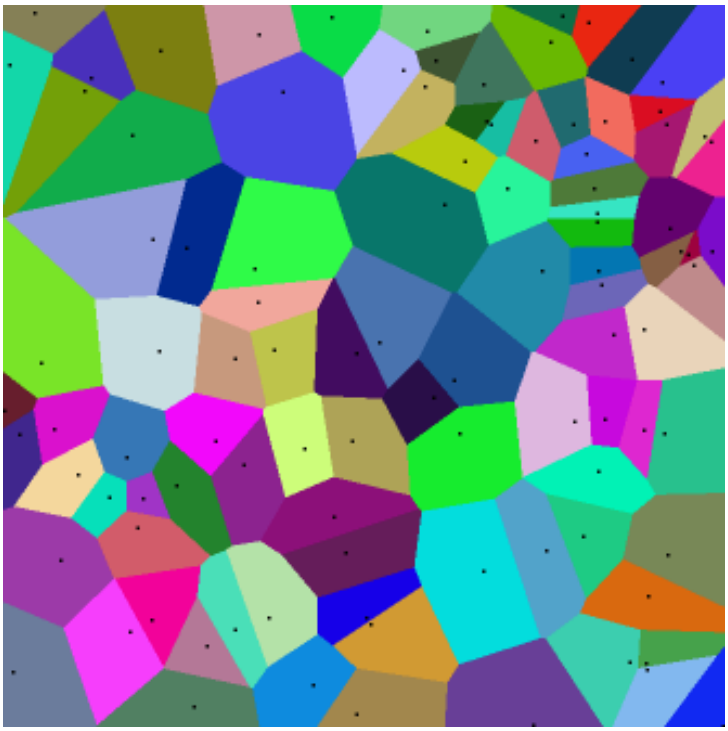
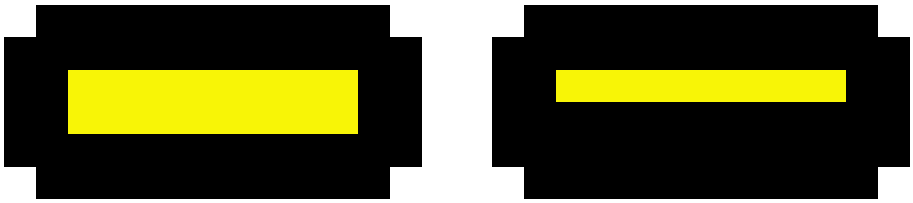


FIGURE 2.10 – *Exemple de diagramme de Voronoï.*

FIGURE 2.11 – *Exemple de squelette de Voronoï. Le bord de la forme est constitué par la ligne rouge, les points échantillonnés sont en bleu et le squelette en noir (Dardenne et al. (2009)).*

échantillonné constitue un germe. Ensuite, le diagramme de Voronoï est calculé de diverses manières (Ogniewicz et Ilg (1992), Ogniewicz et Kübler (1995), Brandt et Algazi (1992), Culver (2000), Milenkovic (1993), Mayya et Rajan (1996)). Le squelette est constitué des arêtes de Voronoï situées à l'intérieur de la forme qui ne sont pas en contact avec le bord. La construction d'un tel squelette est en $O(n \log n)$ pour les méthodes les plus efficaces. Un exemple de squelette de Voronoï est présenté sur la Figure 2.11. Plus la densité de l'échantillonnage est élevée et meilleure est la précision du squelette.

Nous sommes volontairement non-exhaustifs à propos de cette catégorie de méthodes



(a) (b)

FIGURE 2.12 – *Exemple de forme ayant une épaisseur paire (6 pixels) associée à son squelette (en jaune) centré mais épais (a), décentré mais fin (b).*

puisque nous avons choisi de travailler dans le domaine discret, cela n'est donc pas le coeur de notre travail. En revanche, il est intéressant de savoir que l'on peut calculer un squelette à partir du diagramme de Voronoï et de comprendre comment il est obtenu, d'où l'existence de cette partie dans cette thèse.

Conclusion

Dans cette partie, nous avons défini le squelette et présenté les diverses manières de le calculer que l'on peut trouver dans la littérature. Pour être plus exhaustif, Saha et al. (2015) ont dressé une vue générale des algorithmes de squelettisation et leurs applications. En résumé, un squelette idéal devrait avoir ces propriétés :

- finesse (un pixel d'épaisseur) ;
- homotopie à la forme initiale ;
- centré sur la forme ;
- robustesse au bruit ;
- invariance par transformations linéaires ;
- réversibilité (possibilité de reconstruction de la forme initiale).

Néanmoins, dans l'espace discret, certaines propriétés sont en contradiction les unes avec les autres compte tenu des arrondis. En effet, par exemple, on comprend aisément qu'une forme ayant une épaisseur paire, telle que présentée sur la Figure 2.12, aura un squelette de deux pixels d'épaisseur ou un squelette non centré.

La catégorie de méthodes utilisée varie donc en fonction de l'application. La Table 2.1 résume les caractéristiques de chaque catégorie de méthodes.

	fin	homotopie	centré	réversible	géométrie
Amincissement	oui	oui	non	non	discrète
Carte de distance	non	non	oui	oui	discrète
Diagramme de Voronoï	oui	oui	oui	non	continue

TABLE 2.1 – *Caractéristiques des différentes catégories de méthodes de squelettisation présentées précédemment.*

Dans ce tableau, nous remarquons que les squelettes basés sur le diagramme de Voronoï semblent avoir plus de propriétés positives que les deux autres. Ceci est dû au fait qu'il soit calculable dans l'espace continu.

Les propriétés énoncées dans ce tableau sont les propriétés fondamentales de chaque catégorie. Or, il est possible de palier les limites propres à chaque catégorie en ajoutant des post-traitements (Arcelli et Sanniti di Baja (1993), Sanniti di Baja et Thiel (1996)) ou en utilisant des techniques dites "hybrides" (Couprie et al. (2007), Merad et al. (2007), Kimmel et al. (1995)) pour obtenir les propriétés désirées. C'est cette dernière solution que nous avons choisie dans la thèse comme nous le verrons par la suite.

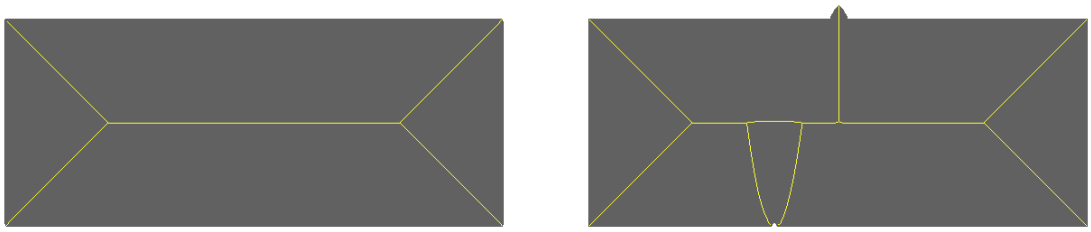
Nous avons également vu que le squelette est un descripteur de forme capable de capturer l'allure générale de la forme ainsi que les détails. Dans certaines utilisations, pour l'appariement de formes notamment, il est nécessaire de déterminer une hiérarchie entre les branches pour savoir celles qui décrivent globalement la forme et celles qui la caractérisent plus localement.

Depuis quelques dizaines d'années, certains chercheurs se sont intéressés à ce problème. La partie suivante présente donc l'état de l'art sur la hiérarchisation du squelette.

2.2 Hiérarchisation et élagage du squelette

L'inconvénient majeur du squelette est sa sensibilité au bruit sur le bord de la forme. Ce bruit est la présence de pixels parasites qui s'ajoutent de façon aléatoire sur le contour de la forme. Une perturbation du bord de l'objet n'entraîne une modification que d'une partie limitée du squelette. Néanmoins, cette modification peut être importante. Les détails du bord qui semblent négligeables visuellement par rapport à l'ensemble de la forme peuvent ainsi générer de longues branches, comme Pizer et al. (1987) l'ont mis en évidence. La longueur des branches n'est pas proportionnelle à la taille de la perturbation dont elle est issue. En d'autres termes, plus une forme est épaisse au niveau de la perturbation et plus la branche issue de cette dernière sera longue pour rejoindre le squelette. Prenons l'exemple de la Figure 2.13, qui représente un rectangle sans bruit et son squelette associé. La Figure 2.13(b) montre ce même rectangle auquel nous avons ajouté, sur le bord, deux perturbations (une excroissance et un creux). La longueur de la branche causée par ces dernières est démesurée par rapport à la taille des déformations. C'est la raison pour laquelle nombre de chercheurs pensent qu'il est nécessaire d'attribuer plus ou moins d'importance aux branches du squelette dans le cas où celui-ci est utilisé, par exemple, pour faire de la reconnaissance de formes planes. La logique veut que si une branche est issue de l'allure générale de la forme alors elle est importante. En revanche, si elle provient d'un détail alors elle est négligeable.

Dans la littérature, les chercheurs appréhendent la hiérarchisation des branches du squelette de diverses manières. Certains utilisent des relations de parenté (Partie 2.2.1), d'autres ont une approche multirésolution (Partie 2.2.2) et quelques-uns sous-entendent la notion de hiérarchie (Partie 2.2.3).



(a) (b)

FIGURE 2.13 – *(a) Squelette d'un rectangle, (b) Squelette du même rectangle auquel a été ajouté une perturbation sur le bord.*

2.2.1 Hiérarchisation par relation de parenté

Dans cette catégorie de méthodes, la hiérarchie est représentée par une relation parent-enfant entre les branches voisines. La branche "parent" étant celle à laquelle il a été attribué la plus grande importance entre les branches voisines. La Figure 2.14 illustre ce concept.

Pour ce faire, Ogniewicz et Kübler (1995) attribuent un degré d'importance à chaque point du squelette. Cette mesure est basée sur le fait que chaque point du squelette est le centre d'un disque tangent à la forme en au moins deux points : $\boldsymbol{p}$ et $\boldsymbol{p}*$. Soit $d^B(\boldsymbol{p}, \boldsymbol{p}*)$ la plus petite distance reliant $\boldsymbol{p}$ à $\boldsymbol{p}*$ obtenue en parcourant le bord de la forme. Plus $d^B(\boldsymbol{p}, \boldsymbol{p}*)$ est importante, moins la branche correspondante est sensible au bruit. Pour illustrer ceci, prenons l'exemple de la Figure 2.15. $d^B(\boldsymbol{p}_A, \boldsymbol{p}_A*)$ est largement supérieure à $d^B(\boldsymbol{p}_B, \boldsymbol{p}_B*)$ ou $d^B(\boldsymbol{p}_C, \boldsymbol{p}_C*)$. Par conséquent, la branche comprenant m_A (centre du disque tangent en $\boldsymbol{p}_A$ et $\boldsymbol{p}_A*$) est plus importante que les branches comprenant m_B ou m_C. Pizer et al. (1987) avaient la même vision de la hiérarchisation du squelette. Pour donner une mesure d'importance à chaque branche, les auteurs remarquent qu'en lissant la forme successivement et en recalculant le squelette sur chaque forme lissée, les branches de celui-ci disparaissent peu à peu. Cependant, il existe encore nombre de problèmes à résoudre, comme la perte de la topologie de la forme lors du lissage ou encore une mauvaise correspondance entre les squelettes successifs.

Une toute autre manière permettant de hiérarchiser le squelette grâce à une relation de parenté est l'utilisation des graphes de choc (shock graphs). Cette notion a été développée par Siddiqi et al. (1999) et utilisée par Sebastian et al. (2004), notamment. Le système des chocs provient du processus d'évolution de la courbe représentant le bord de la forme tel que le feu de prairie de Blum (1967). Autrement dit, un graphe de choc est une interprétation dynamique d'un squelette dans lequel les chocs sont des points de singularité formés par la propagation des ondes à partir du bord, perpendiculairement à ce dernier et à une vitesse constante. Il existe quatre types de choc. Pour les illustrer, nous associons chaque type de choc à la variation locale de la transformée en distance $\hat{d}_E$ (définie au chapitre 1) le long du squelette. Un point de squelette s est un choc de :

– **premier ordre** : $\hat{d}_E$ varie de façon monotone au voisinage de s (*cf.* Figure 2.16(a)).

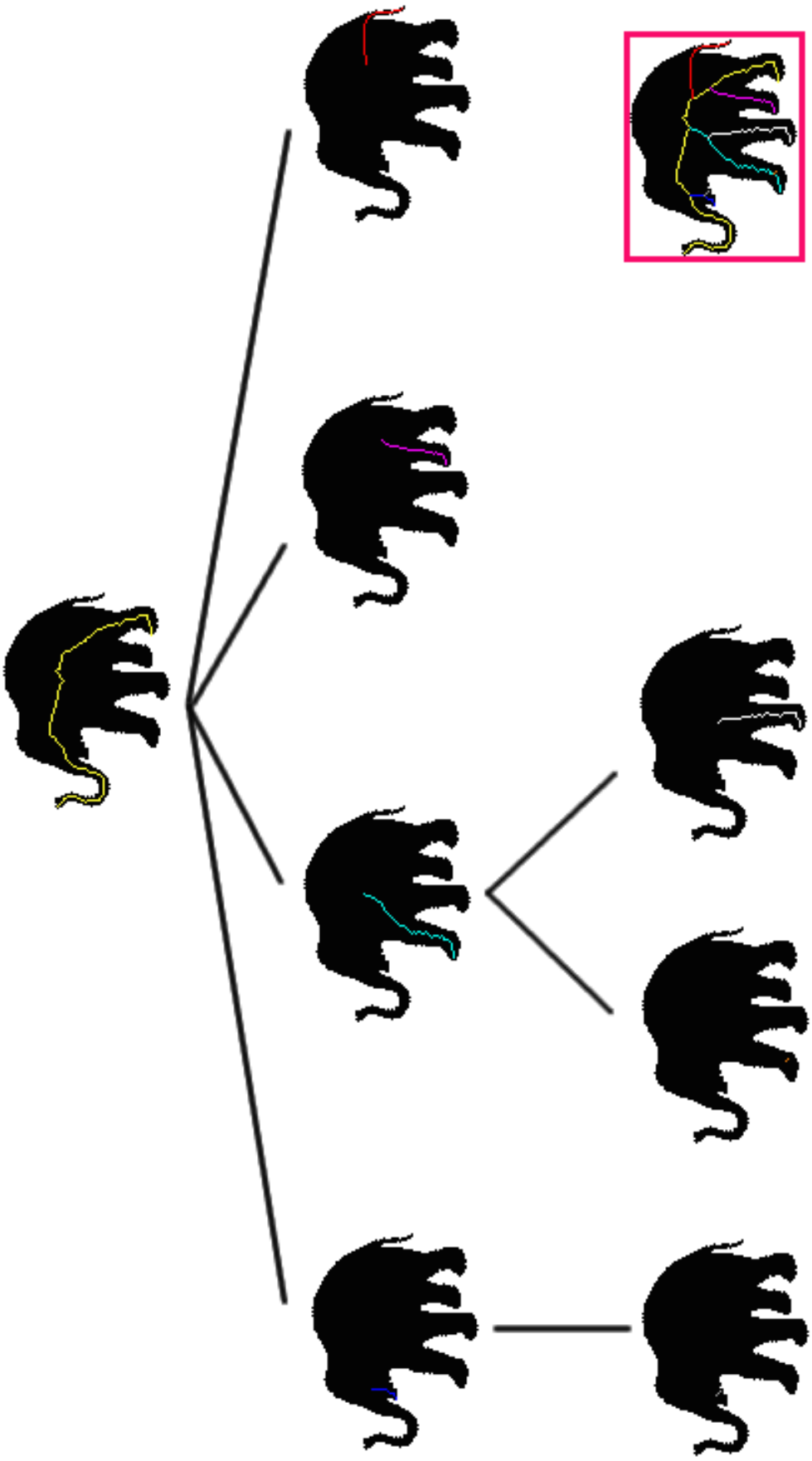


FIGURE 2.14 – *Illustration de la relation parent-enfant entre les branches pour le squelette de la forme encadrée en rouge.*

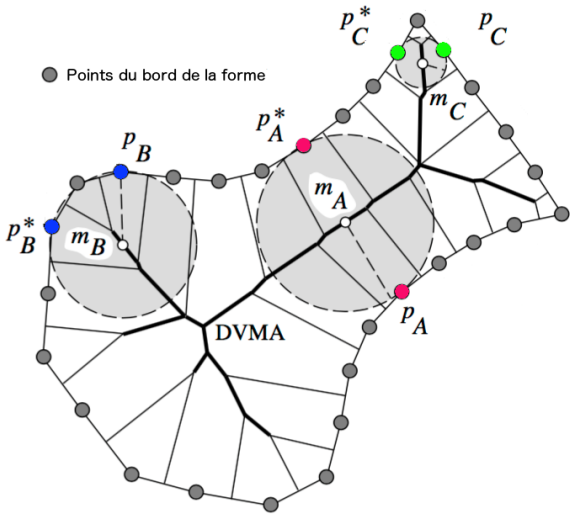


FIGURE 2.15 – *Illustration de l'importance des branches pour Ogniewicz et Kübler (1995).*

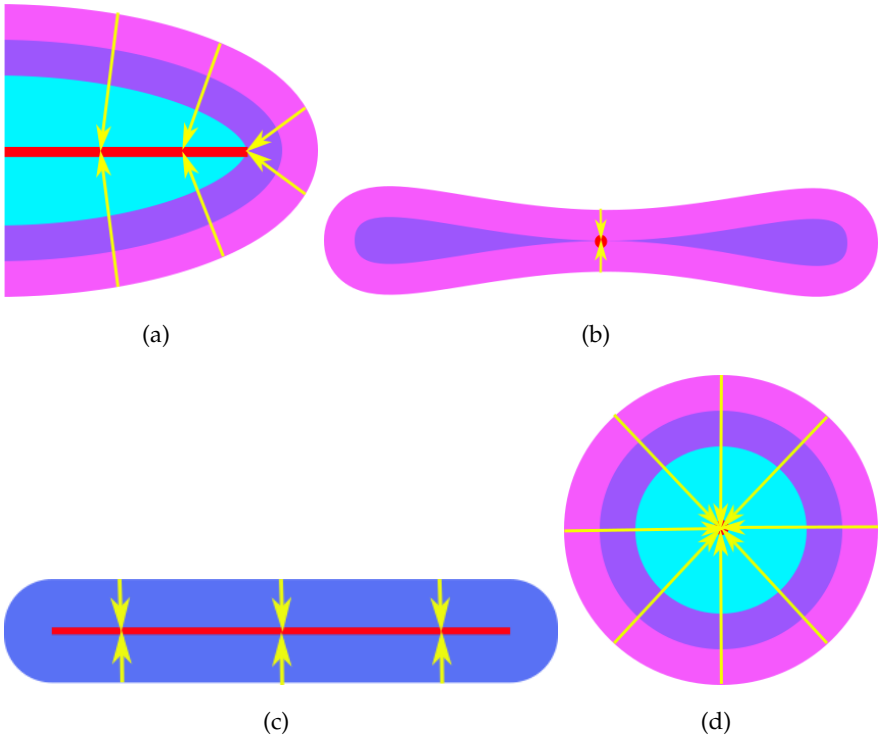
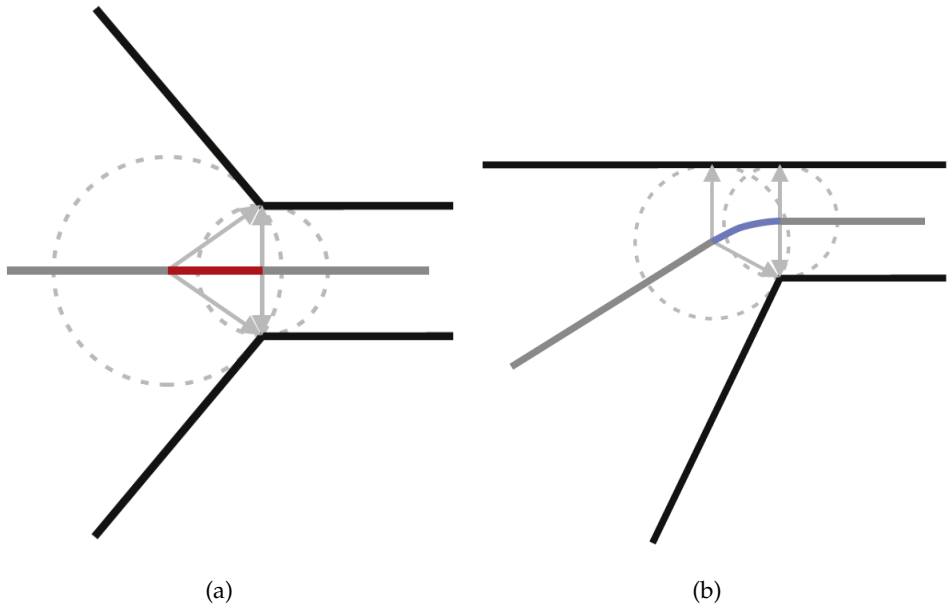


FIGURE 2.16 – *(a) Choc du premier ordre, (b) Choc du deuxième ordre, (c) Choc du troisième ordre et (d) Choc du quatrième ordre. Les chocs sont représentés en rouge.*

- **deuxième ordre** : s est un minimum local de $\hat{d}_E$. C'est le cas quand s est situé sur un goulet (*cf.* Figure 2.16(b)).
- **troisième ordre** : $\hat{d}_E$ est constante au voisinage de s (*cf.* Figure 2.16(c)).
- **quatrième ordre** : s est un maximum local de $\hat{d}_E$ (*cf.* Figure 2.16(d)).

La construction d'un graphe acyclique orienté (DAG pour Directed Acyclic Graph) est réalisable en considérant que chaque nœud est formé par un choc de deuxième ou quatrième



(a) (b)

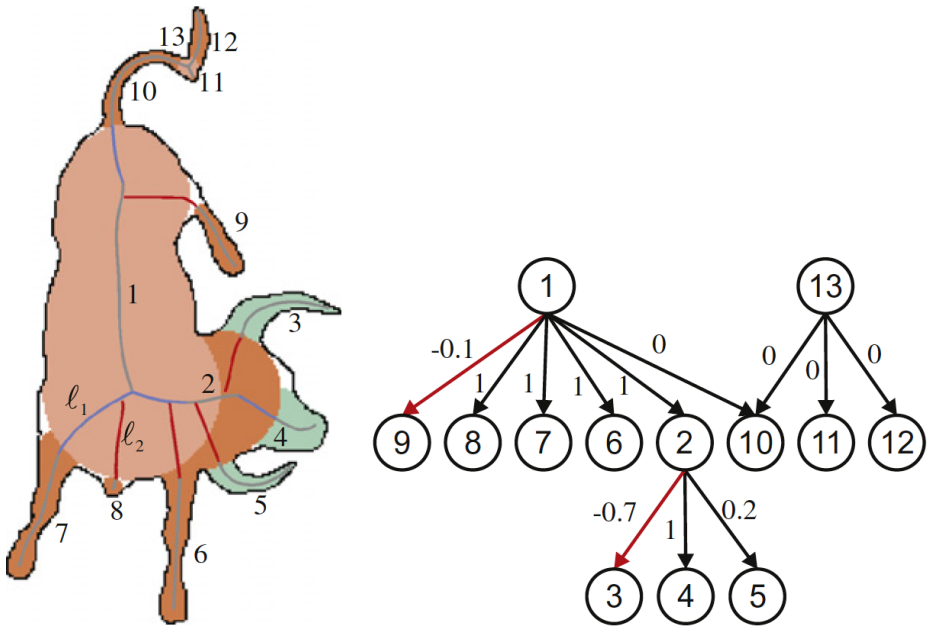
FIGURE 2.17 – *(a) Ligature pleine (en rouge), (b) Semi-ligature (en bleu).*

ordre et que chaque arête est créée par une séquence continue de chocs de premier et troisième ordre.

Plus récemment, Macrini et al. (2008; 2011a;b) ont travaillé sur les graphes d'os (bone graphs). Selon les auteurs, l'intégralité du squelette n'est pas nécessaire pour encoder la forme. En effet, certaines parties de branches codent les parties saillantes du bord de la forme. En revanche, les concavités du bord situées de chaque côté des parties saillantes sont responsables de points spéciaux du squelette appelés ligatures dans lesquelles les points de squelette sont groupés par segment. Soit s un point de squelette et, $\boldsymbol{p}_s$ et $\boldsymbol{p}_s^*$, les deux points de contour tangents à la boule maximale centrée en s. Il existe deux types de ligature :

- **ligature pleine** : ensemble de points connectés s d'une branche de squelette tels que $\boldsymbol{p}_s$ et $\boldsymbol{p}_s^*$ sont tous deux des concavités du contour. Ce type de ligature est représenté en rouge sur la Figure 2.17(a).
- **semi-ligature** : ensemble de points connectés s d'une branche de squelette tels que seul $\boldsymbol{p}_s$ ou $\boldsymbol{p}_s^*$ est une concavité du contour. Ce type de ligature est représenté en bleu sur la Figure 2.17(b).

Étant donné que les pixels du squelette appartiennent à une ligature ou codent les parties saillantes de la forme, pour créer un DAG, les auteurs ont choisi des noms parlants qui font référence à l'anatomie. Un os est associé à une protubérance sur le bord de la forme et un ligament relie deux os. Un ligament est donc une arête orientée dont la création est basée sur la configuration des ligatures. L'attribut de chaque arête représente la position où l'os enfant est relié à l'os parent. La figure 2.18 représente une vache sur laquelle les ligatures sont mises en évidence ainsi que le DAG associé (graphe d'os).



(a) (b)

FIGURE 2.18 – *(a) Squelette d'une vache mettant en évidence les ligatures, (b) Graphe d'os de la vache.*

2.2.2 Hiérarchisation par approche multirésolution

La hiérarchie peut également être vue comme une séquence de squelettes, dans laquelle chaque squelette correspond à une représentation de la forme à une certaine échelle. Ainsi, il existe des squelettes permettant de reproduire la forme grossièrement, et d'autres permettant de la reconstruire dans les moindres détails. Un des articles de la littérature concernant ce sujet est écrit par Borgefors et al. (2001), qui obtiennent une séquence de squelettes en travaillant sur les points du squelette. Les auteures diminuent la résolution de l'image successivement. La taille des images étant de plus en plus petite, les détails disparaissent peu à peu. Le squelette est recalculé sur chaque image et les parties de celui-ci correspondant à la même partie de la forme sont appariées. Ainsi, plus les parties de squelettes persistent et plus elles sont importantes et plus les points de squelettes associés sont considérés comme importants. Les auteures précisent également que les points du squelette d'origine peuvent être pondérés par niveau de résistance au fur et à mesure du rétrécissement de l'image (*cf.* Figure 2.19 pour laquelle les points les plus importants sont les rouges puis les verts, les bleus et enfin les jaunes).

Par ailleurs, une des dernières méthodes permettant de hiérarchiser un squelette grâce à une approche multirésolution a été écrite par Yang et al. (2016). Les auteurs ne travaillent pas sur les points de squelette mais sur l'importance des branches. Ils utilisent une représentation polygonale du contour de la forme, dans laquelle chaque segment est une approximation d'une partie du contour et chaque sommet convexe est l'origine d'une branche de squelette (Bai et al. (2007)). Ils simplifient ce polygone au fur et à mesure en supprimant les sommets un par un en fonction de la longueur des deux segments qui en sont issus et de l'angle qu'ils forment. Lorsqu'un sommet est supprimé, ses deux sommets voisins sont reliés par un segment. Par conséquent, la branche de squelette issue du sommet supprimé est évincée, ce qui

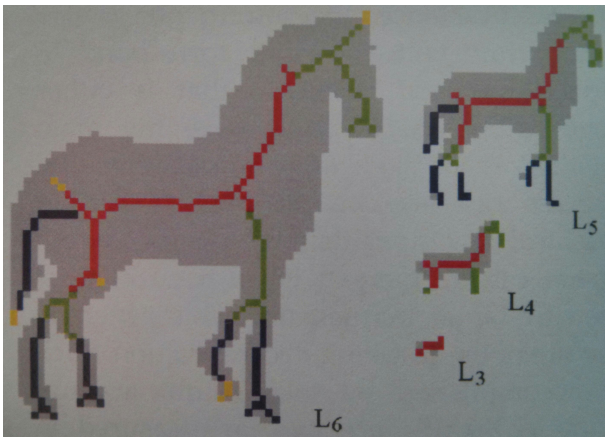


FIGURE 2.19 – *Illustration de la hiérarchie de Borgefors et al. (2001).*

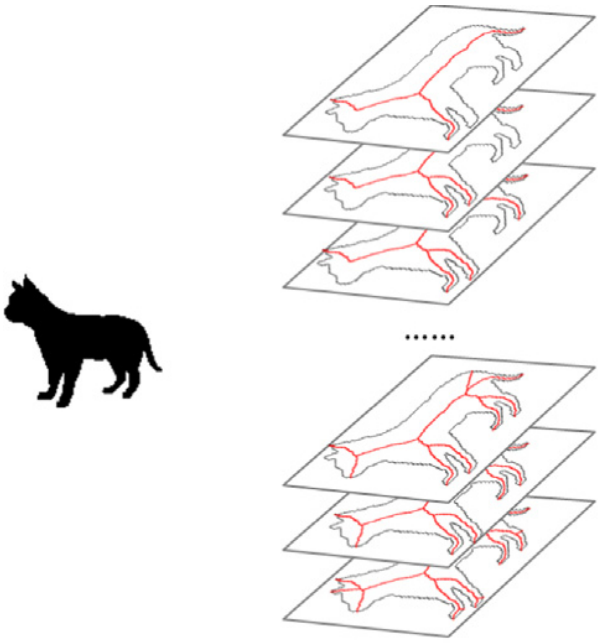


FIGURE 2.20 – *Illustration d'une séquence de squelettes hiérarchisée Yang et al. (2016).*

crée un squelette ajouté à la séquence sus-citée. Un exemple de séquence de squelettes est présenté sur la Figure 2.20.

2.2.3 Hiérarchisation sous-entendue

Certains auteurs sous-entendent la notion de hiérarchie comme Chazal et Lieutier (2005), Chaussard et al. (2011) qui définissent la *weak feature size (wfs)* comme la distance minimum entre $\bar{\mathcal{F}}$ et les points critiques de la fonction de distance de $\mathcal{F}$. Ainsi, ils créent le λ-axe médian qui est un sous-ensemble du squelette de $\mathcal{F}$ "capturant" l'homotopie de $\mathcal{F}$ quand $\lambda < wfs$. En faisant varier λ, ils obtiennent une hiérarchie dans le sens où ils sont capables d'avoir une séquence de squelettes plus ou moins grossiers (avec plus ou moins de détails).

Par ailleurs, Shen et al. (2011) ont créé le Rapport de Potentiel de Flexion (RPF) pour donner une valeur d'importance à chaque branche. Cette mesure est basée sur l'importance des segments de contours en tenant compte de l'information contextuelle. Soit $\boldsymbol{p}$ un point du sque-

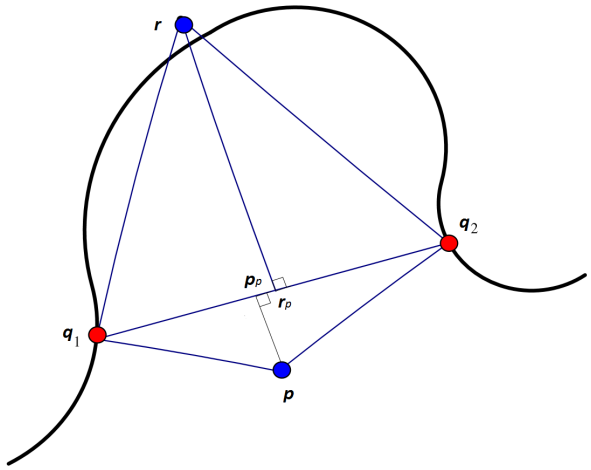


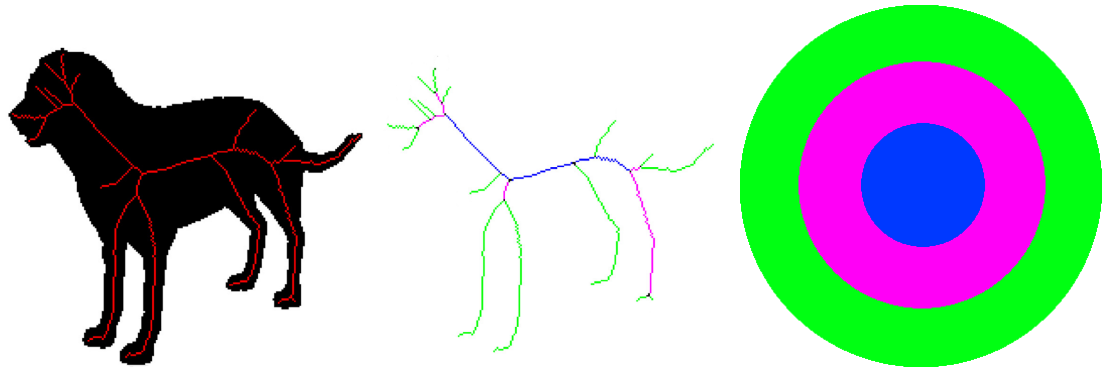
FIGURE 2.21 – *Illustration des notations introduites pour calculer l'importance d'un point de squelette pour Shen et al. (2011).*

lette, q_1 et q_2 deux points du bord les plus proches de p pour lesquels $d(p, q_1) = d(p, q_2)$. $r \in \mathbb{R}^2$ est appelé point fantôme de p. Il est obtenu en construisant un triangle isocèle ayant pour base le segment $[q_1, q_2]$ tel que $d(r, q_1) = d(r, q_2) = \frac{1}{2} l(q_1,\ q_2)$ où $l(q_1,\ q_2)$ est la longueur du contour entre q_1 et q_2. Notons r_p (respectivement p_p) le projeté orthogonal de r (respectivement p) sur $[q_1, q_2]$. Le RPF est noté $\varepsilon(p, q_1, q_2) = \frac{d(r, r_p)}{d(p, p_p)}$. Plus la valeur obtenue est grande et plus le point de squelette p est important. Toutes les notations introduites sont précisées sur la Figure 2.21.

Montero et Lang (2012) ne parlent pas non plus de hiérarchie mais sont capables de donner des squelettes plus ou moins simplifiés en fonction d'un seuil. Nous pouvons donc considérer ceci comme une hiérarchie de squelettes. Ils basent également leur méthode sur une approximation du contour (Bai et al. (2007)), puis ils sélectionnent les pixels du squelette ayant des points de bord équidistants sur deux côtés de la forme.

De même, Yang et al. (2016), Liu et al. (2012; 2013) utilisent une simplification de la forme par un polygone (Bai et al. 2007) pour déterminer les points du bord de la forme les plus saillants (points du bord de la forme pour lesquels la courbure est convexe et maximale localement). En d'autres termes, il s'agit des points potentiellement utiles pour supprimer le bruit. Pour savoir si une branche issue d'un point saillant peut être supprimée sans modifier l'allure générale de la forme, les auteurs utilisent une mesure d'importance à la fois globale (pourcentage de reconstruction de la branche utilisé par Bai et Latecki (2007), Attali et al. (1995)) et locale (longueur de la partie de la branche qui reconstruit à elle seule une partie de la forme). Une branche ayant une faible importance est une branche pour laquelle les mesures d'importance locales et globales sont faibles.

Quant à Serino et Sanniti di Baja (2015), ils évaluent l'importance de chaque branche en prenant en compte la perte entre la forme initiale et la forme reconstruite à partir du squelette auquel on supprime successivement des branches. Les mesures d'importance ne sont calculées que sur le squelette initial et permettent une hiérarchie en couronne comme présentée sur la Figure 2.22. À chaque niveau de hiérarchie, les branches du squelette, qui peuvent



(a) (b) (c)

FIGURE 2.22 – *(a) Squelette d'un chien, (b) squelette hiérarchique associé au chien, (c) schématisation de la hiérarchisation en couronne.*

être interprétées comme des branches périphériques à ce niveau, sont temporairement supprimées et forment une couronne. Cette suppression de couronne par ordre d'importance permet à chaque branche de pouvoir devenir une branche périphérique. Ainsi, plus les branches font partie d'une couronne extérieure moins elles sont importantes. Lorsque chaque branche appartient à une couronne, la concaténation des branches allant de la couronne la plus importante vers la moins importante permet d'obtenir une hiérarchie en respectant la topologie pour chaque niveau. Ainsi, un squelette peut être élagué plus ou moins sévèrement en cessant de concaténer les couronnes situées vers l'extérieur.

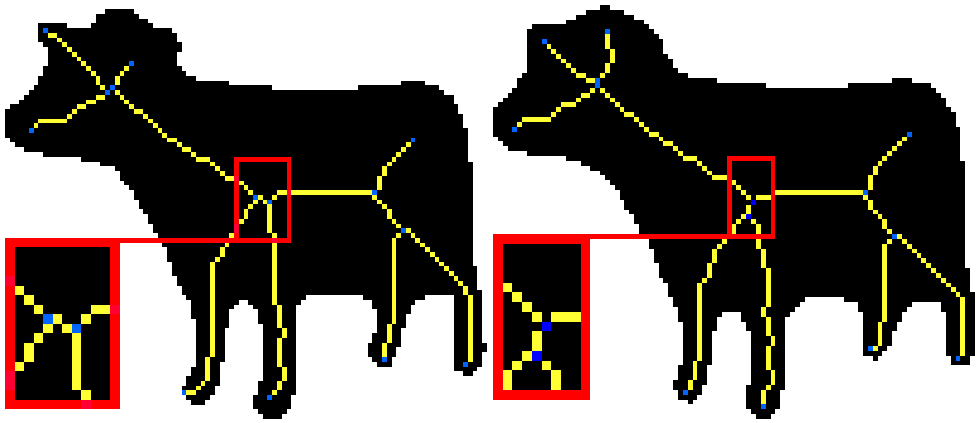
L'élagage (ou l'ébarbulage) permettant de simplifier le squelette pour qu'il soit plus robuste au bruit est une conséquence de sa hiérarchisation. La partie suivante décrit les approches d'ébarbulage existantes.

2.2.4 Une conséquence : l'Élagage

Comme expliqué précédemment, les méthodes de squelettisation sont généralement sensibles au bruit, ce qui crée des branches inintéressantes dans le squelette. Certains chercheurs choisissent de leur donner une importance négligeable (hiérarchisation), d'autres préfèrent supprimer définitivement les branches qu'ils considèrent inutiles afin d'accroître la stabilité au bruit du squelette (ébarbulage). L'élagage du squelette vise donc à supprimer les branches causées par le bruit sur le contour de la forme (celles qui ont le moins de signification) tout en respectant la topologie de la forme. Les méthodes peuvent être divisées en deux catégories.

La première consiste à lisser le bord de la forme avant de calculer le squelette dit ébarbulé (Mokhtarian et Mackworth (1992), Shaked et Bruckstein (1998; 1996)). La plupart de ces méthodes sont relativement anciennes et présentent un problème majeur : la production d'un biais dans le squelette dû à la modification de la forme. Par 'biais', nous entendons le déplacement des branches par rapport au squelette obtenu sur la forme initiale. Il est mis en évidence sur la Figure 2.23.

La seconde regroupe des méthodes plus récentes dont le but est de réaliser un post-traitement sur le squelette en attribuant des valeurs d'importance à chaque branche. En



(a) (b)

FIGURE 2.23 – *(a) Squelette d'une forme initiale de vache, (b) Squelette biaisé sur la forme de la vache légèrement lissée.*

d'autres termes, il s'agit de hiérarchiser le squelette puis de l'élaguer branche par branche. Le principe de l'élagage est de prendre une décision binaire concernant la conservation d'une branche en fixant un seuil minimal d'importance. Toutes les méthodes que nous avons présentées dans les Parties 2.2.1, 2.2.2 et 2.2.3 peuvent permettre la suppression des parties du squelette ayant une valeur d'importance inférieure au seuil fixé pour ne conserver que les branches importantes et avoir un squelette plus robuste au bruit (Borgefors et al. 2001, Yang et al. 2016, Bai et al. 2007, Liu et al. 2012; 2013, Montero et Lang 2012, Shen et al. 2011, Serino et Sanniti di Baja 2015).

2.3 APPARIEMENT DU SQUELETTE

LE squelette soulève deux difficultés lorsqu'il est utilisé pour l'appariement de formes :

- il est sensible au bruit
- les squelettes respectifs de deux formes de la même catégorie peuvent avoir une structure différente, comme illustré en Figure 2.24

Dans la reconnaissance de formes basée sur le squelette, les graphes sont majoritairement utilisés car ils présentent l'avantage d'être invariants à la translation, à la rotation et à la symétrie. Ainsi, le problème d'appariement de squelettes devient très souvent un problème d'appariement de graphes.

Dans la littérature, il existe deux grandes catégories d'appariement de graphes :

1. Appariement de graphes exact
2. Appariement de graphes inexact

Cependant, la présence de bruit sur les formes représentant des objets réels est quasi-systématique. De plus, deux formes, et par conséquent leurs squelettes respectifs, peuvent se ressembler sans pour autant être identiques. Ainsi dans le cadre de notre utilisation, l'appariement de graphes exact est très peu utilisé pour les formes planes puisque cela reviendrait

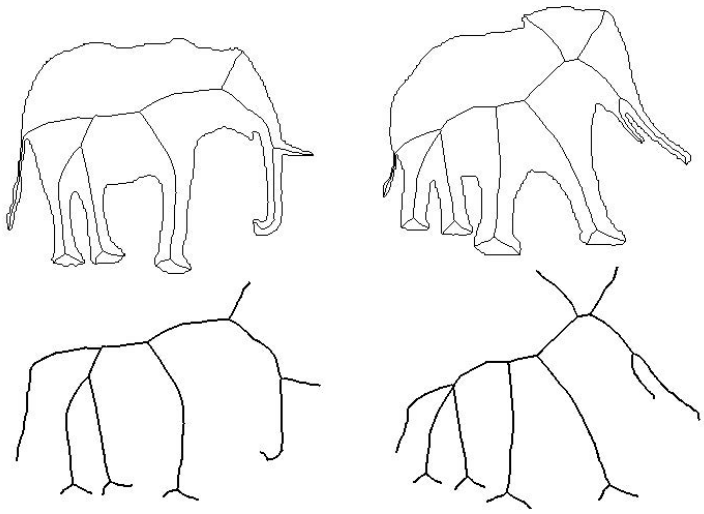


FIGURE 2.24 – *Illustration de la différence de structure du squelette de deux formes faisant partie de la même catégorie (Giang et al. (2013b)).*

à dire que deux graphes sont identiques ou pas. La majorité des méthodes utilise donc un appariement de graphes inexact dans lequel la tolérance à l'erreur fait partie du processus.

Dans cette partie, nous nous focalisons volontairement sur les méthodes d'appariement utilisant spécifiquement le squelette d'une forme. Néanmoins, pour avoir une représentation plus large de l'appariement appliqué à la reconnaissance de modèles dans sa globalité (images, signaux audios/vidéos, composés chimiques, réseau métabolique), nous pouvons nous référer aux travaux de Livi et Rizzi (2013), qui fait un état de l'art détaillé dans ce domaine.

Pour accroître la qualité des résultats, la plupart des méthodes tiennent comptent de la hiérarchie entre les branches du squelette. Dans la littérature, il existe deux façons principales de concevoir la hiérarchisation de squelettes (*cf.* Partie 2.2) grâce aux graphes. La hiérarchisation des branches peut être représentée :

- par les arcs dans un graphe orienté : le sens de l'arc induit une hiérarchie.
- par les valeurs dans un graphe attribué : une valeur représente le niveau hiérarchique de la branche.

Le squelette peut donc être modélisé par différents types de graphes. Nous avons choisi d'expliquer les méthodes utilisées en fonction du type de graphe créé.

2.3.1 Graphe acyclique orienté (DAG pour *Directed Acyclic Graph*)

Il s'agit d'un objet qui se compose de nœuds et d'arcs, dans lequel chaque arc est dirigé d'un nœud à un autre de manière à ce que l'on ne puisse pas trouver une séquence d'arcs qui parte d'un nœud et qui revienne à son point de départ. En résumé, les arcs orientés ne doivent pas former de cycle.

Distance d'édition de graphes (GED)

Utilisée à l'origine sur des chaînes de caractères (Levenshtein (1966)), le principe de la GED est de définir un coût pour chaque opération d'édition basique puis de déterminer l'ensemble

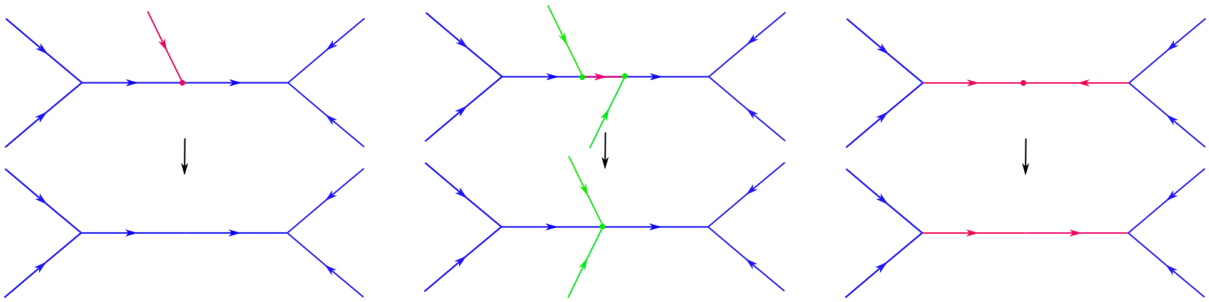


FIGURE 2.25 – *Trois opérations d'édition du graphe de chocs, (a) Jointure, (b) Contraction et (c) Fusion.*

fini des opérations d'édition permettant de transformer un graphe en un autre en minimisant le coût total des opérations utilisées. La GED peut être appliquée sur divers types de graphes (voir l'état de l'art de Gao et al. (2010)). Elle est en particulier utilisée sur les graphes de chocs (Siddiqi et al. (1999), Sebastian et al. (2004)) que nous avons présentés dans la Section 2.2.1. Nous rappelons que les arcs sont constitués des chocs d'ordre un et trois dans lesquels les temps de formation des chocs dirigent les nœuds alors que les nœuds sont des chocs d'ordre deux et quatre.

Le **degré** d'un nœud est le nombre d'arcs issus de ou dirigés vers ce nœud.

Il existe quatre opérations de base dans les graphes de chocs :

1. jointure : l'opération retire un arc associé à un nœud feuille (de degré 1) et fusionne les deux arcs adjacents à celui-ci (Figure 2.25(a)).
2. contraction : l'opération supprime un arc connectant deux nœuds de degré 3 (Figure 2.25(b)).
3. fusion : l'opération combine deux arcs reliés à un nœud de degré 2 (Figure 2.25(c)).
4. déformation : l'opération apparie deux arcs ayant des attributs différents.

Le problème majeur de la *GED* est que la flexibilité de cette catégorie de méthodes implique un coût de calcul important lorsqu'aucune contrainte n'est imposée sur la structure du graphe ou qu'il n'existe pas d'attribut associé aux nœuds ou arcs. En effet, les algorithmes sont souvent basés sur une procédure de recherche combinatoire qui explore l'espace de toutes les transformations du graphe valides (Sanfeliu et Fu 1983). Néanmoins, il existe des méthodes permettant d'approximer la distance d'édition de graphe comme Fischer et al. (2015).

Appariement récursif

Globalement, ce mécanisme consiste à construire itérativement une solution en ajoutant les appariements de nœuds compatibles avec ceux déjà présents dans la solution partielle. La recherche est généralement guidée par une valeur de similarité. Plus précisément, étant donnés deux DAGs, l'idée générale est d'apparier récursivement les nœuds. Pour ce faire, Shokoufandeh et al. (2005; 2006) calculent un graphe biparti ayant, d'une part, les nœuds du premier DAG ($DAG^{(1)}$) et, d'autre part, les nœuds du second DAG ($DAG^{(2)}$). Ils créent un

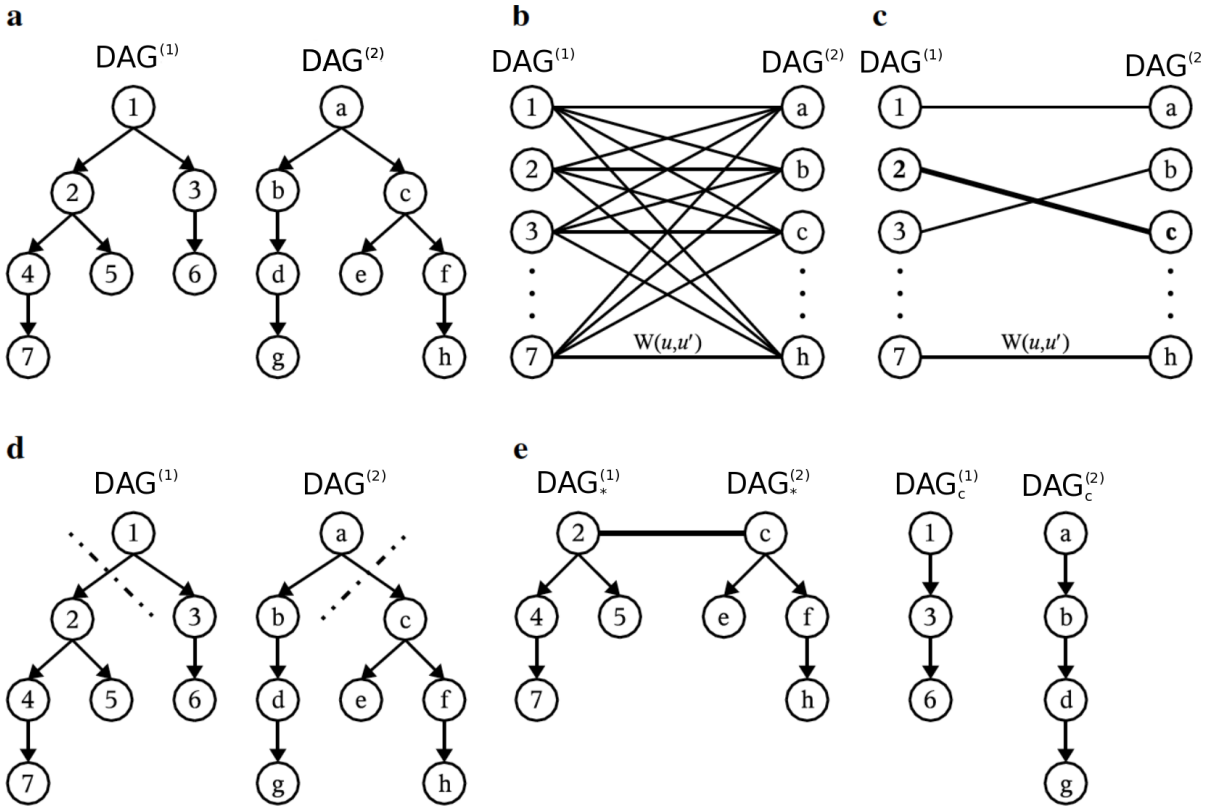


FIGURE 2.26 – *Illustration de l'appariement de deux DAGs. (a) Deux DAGs, (b) Création d'un graphe biparti associé à la pondération des arêtes suivant la valeur de similarité entre les nœuds, (c) Calcul de l'appariement maximum et ajout du meilleur à l'ensemble des solutions, (d) Scission des graphes au niveau des nœuds appariés, (e) Utilisation du même processus récursivement (Shokoufandeh et al. 2006).*

arc entre chaque paire de nœuds (u, u'), u appartenant à $DAG^{(1)}$ et u' à $DAG^{(2)}$, ayant pour poids la valeur de similarité entre u et u'. Les nœuds sont associés deux à deux par ordre croissant des poids et sont ajoutés à la solution au fur et à mesure. La procédure est appliquée récursivement aux deux DAGs ayant pour racine les deux nœuds nouvellement appariés. Cet algorithme est détaillé sur la Figure 2.26.

Cet algorithme a notamment été utilisé sur les graphes d'os que nous avons décrits dans la Partie 2.2.1 (Macrini et al. (2008; 2011a;b)) dans lesquels les nœuds représentent les os et les arcs, les ligaments.

2.3.2 Graphe Relationnel Attribué (ARG)

Un ARG est un graphe dans lequel chaque sommet et/ou chaque arête possède un attribut permettant une meilleure caractérisation. Certains chercheurs ne sont pas intéressés par le fait d'apparier chaque sommet à un et un seul appartenant à un autre graphe puisqu'ils recherchent uniquement une valeur de dissimilarité entre deux graphes. Pour ce faire, ils s'autorisent à apparier plusieurs sommets d'un premier graphe à plusieurs sommets d'un autre graphe, ce qu'on appellera un appariement many-to-many.

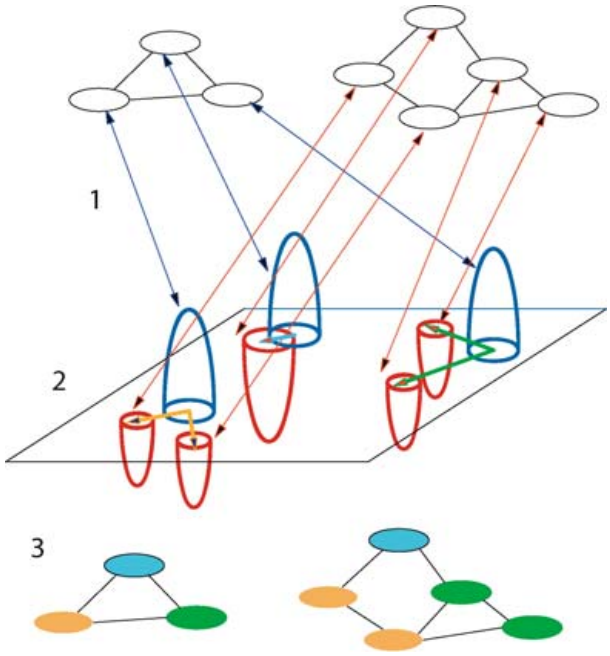


FIGURE 2.27 – *Aperçu de la méthode de Demirci et al. (2006).*

Appariement many-to-many

Étant donné deux ARGs dans lesquels les sommets possédant un attribut sont reliés par des arêtes attribuées également. La méthode de Keselman et al. (2003), Demirci et al. (2006) consiste à placer les deux ARGs dans le même espace géométrique le plus petit possible. Dans cet espace, deux sommets attribués d'un même graphe sont séparés réellement par la distance contenue par l'attribut de chaque arête (représentation par l'étape 1 de la Figure 2.27). Supposons que chaque sommet du premier graphe représente un tas de terre, ayant comme volume son attribut, et chaque sommet du second graphe soit un trou, ayant également comme volume son attribut (*cf. Section 5.1.1*). La distance euclidienne entre un tas et un trou représente la distance à parcourir pour transporter la terre du tas vers le trou (représentation par l'étape 2 de la Figure 2.27). Le problème revient donc à minimiser le travail à réaliser pour déplacer la terre de l'ensemble des tas vers l'ensemble des trous en autorisant le remplissage de plusieurs trous avec une partie d'un tas (représentation par l'étape 3 de la Figure 2.27). Cet algorithme est également appelé "distance du cantonnier". Il est décrit en détail dans le Chapitre 5 car il s'agit d'une des étapes de notre appariement. Le résultat permet donc de donner une mesure de dissimilarité entre deux ARGs sans associer les sommets deux à deux. La Figure 2.27 illustre l'ensemble de ces étapes.

Algorithme Hongrois

Dans leurs travaux, Bai et Latecki (2008), Shen et al. (2013) calculent une valeur de dissimilarité entre chaque paire de plus court chemin reliant deux feuilles de squelette. La manière dont ils procèdent exactement est précisée dans la Partie 2 du Chapitre 5. Chaque plus court chemin peut être modélisé par un sommet dans un ARG biparti complet dans lequel une partie des sommets provient de la première forme et l'autre, de la deuxième forme. Des sommets dits

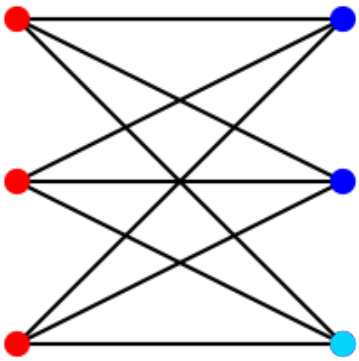


FIGURE 2.28 – *Graphe biparti complet dans lequel chaque sommet rouge est associé à la première forme et chaque sommet bleu, à la deuxième. Chaque sommet rouge est relié à chacun des sommets bleus. Le sommet bleu ciel est un sommet fantôme.*

"fantômes" sont ajoutés dans la partie des sommets étant en infériorité numérique pour qu'il y ait autant de sommets dans chaque partie. La Figure 2.28 permet de visualiser cette notion.

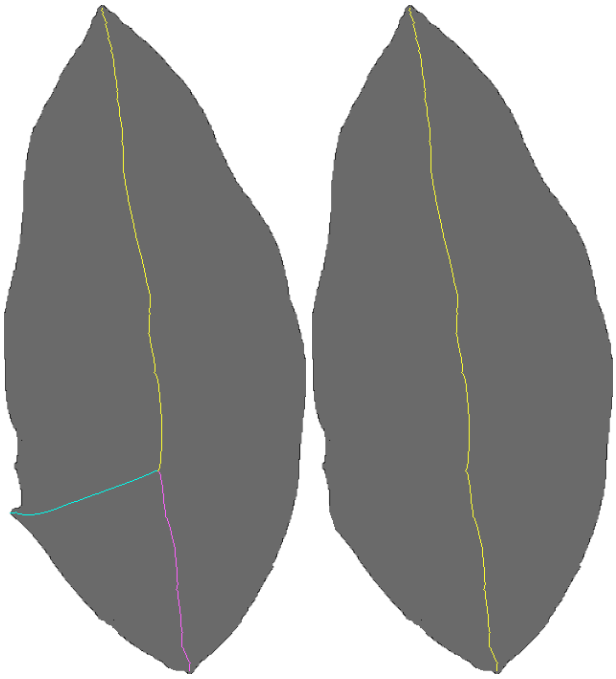
Dans ce graphe, chaque arête a pour attribut la valeur de dissimilarité calculée entre les deux sommets qu'elle relie. Un ARG est alors construit. Les auteurs ont choisi d'appliquer l'algorithme hongrois afin d'associer à chaque sommet un et un seul autre sommet. Cette méthode, que nous utilisons, est détaillée dans la Partie 1 du Chapitre 5.

Marche aléatoire

Giang et al. (2013a;b) construisent un ARG biparti comme dans l'algorithme Hongrois. Les auteurs ne recherchent pas un appariement optimal des sommets mais une valeur minimale de dissimilarité. Pour ce faire, lorsqu'ils obtiennent la valeur de dissimilarité entre deux sommets, ils n'appliquent pas l'algorithme hongrois comme précédemment mais une marche aléatoire repondérée, développée par Cho et al. (2010), qui est une heuristique. Dans l'algorithme de marche aléatoire standard, le marcheur visite successivement les sommets en suivant les arêtes de manière aléatoire. Dans une marche aléatoire repondérée, le marcheur a le droit de se téléporter. Ainsi, il se déplace en parcourant une arête avec une probabilité p ou en se téléportant vers certains sommets contraints avec une probabilité $(1 - p)$. De cette manière, p représente le biais entre deux actions possibles (en suivant une arête ou en se téléportant).

2.3.3 Appariement par la construction d'un arbre itérativement

Certaines méthodes utilisent l'arbre (graphe sans cycle) comme structure de données, non pas pour représenter les données issues des squelettes mais pour construire la solution itérativement. Lors de l'appariement de branches entre deux squelettes, une branche peut être appariée à une autre mais également à un chemin de branches, c'est-à-dire à une concaténation de branches n'ayant que deux extrémités. Ceci permet de modéliser le fait que lorsqu'une zone saillante est produite sur le bord de la forme, une branche y est associée (branche turquoise sur la Figure 2.29(a)) et coupe une autre branche (branche jaune sur la Figure 2.29(b)),



(a) (b)

FIGURE 2.29 – *(a) Squelette d'une feuille, (b) Squelette de cette même feuille privé de la zone saillante responsable de la branche turquoise.*

qui se sépare en deux à cause de la branche provenant de la zone saillante (branches fuschia et jaune sur la Figure 2.29(a)).

Séparation et évaluation (branch and bound)

Le principe consiste à construire un arbre de solutions partielles ayant comme racine le problème initial puis à "diviser pour régner". Pour diminuer le nombre prohibitif de solutions à tester, le *branch and bound* énumère ces solutions de manière logique dans le sens où il est possible d'éliminer des solutions partielles qui ne mènent pas à une solution satisfaisante.

Pour ce faire, deux phases sont à considérer récursivement :

1. *Phase de séparation* : division d'un problème en sous-problèmes.
2. *Phase d'évaluation* : décision quant à la conservation de la solution partielle courante. Si elle est déjà moins bonne que la meilleure solution trouvée jusqu'à présent ou qu'une borne supérieure (coût d'une solution réalisable déterminée à l'aide d'une heuristique), on coupe la branche, c'est-à-dire qu'on n'explore pas les solutions filles (sous-problèmes de cette solution partielle courante). Le sous-problème courant est alors éliminé.

Notons que la performance de cette méthode dépend notamment de la qualité de la borne supérieure pour exclure des solutions partielles le plus tôt possible.

Dans le cas du squelette, le problème principal est d'apparier deux squelettes qu'on divise pour tester l'appariement de branches ou concaténation de branches ((Zhu et Yuille 1996)).

A*

Contrairement à la procédure par séparation et évaluation, qui explore implicitement tout l'espace des solutions, A* est une heuristique, c'est-à-dire une méthode qui fournit rapidement une solution réalisable mais pas nécessairement optimale. À chaque itération, on va tenter de se rapprocher de la meilleure solution, on va donc privilégier les possibilités qui donnent directement une meilleure solution, en mettant de côté toutes les autres. La racine virtuelle de l'arbre de solutions est un appariement de coût zéro et ses fils sont l'ensemble des possibilités d'appariement branche/branche ou branche/concaténation de branches. Pour résoudre ce problème, il faut commencer par se diriger vers le coût d'appariement le plus faible. Toutes les possibilités ne permettant pas de se rapprocher de la meilleure solution sont mises de côté, mais pas supprimées pour éventuellement les explorer si jamais la solution choisie s'avère mauvaise. En effet, il est difficile de savoir à l'avance si une solution va aboutir ou sera meilleure qu'une autre. Cet algorithme est utilisé par Liu et Geiger (1999).

2.3.4 Appariement tenant compte du contexte

Pour calculer la mesure de dissimilarité entre deux formes, certains auteurs, comme Bai et al. (2010), l'apprennent en fonction du contexte. Les formes représentant le même objet dans une base de données sont associées à un ensemble de formes $\mathcal{G} = \{\mathcal{F}^{(\mathcal{G}_1)}, \mathcal{F}^{(\mathcal{G}_2)}, \ldots, \mathcal{F}^{(\mathcal{G}_{|\mathcal{G}|})}\}$. Pour une forme requête donnée $\mathcal{F}^{(r)}$, la similarité avec $\mathcal{F}^{(\mathcal{G}_i)}$ est apprise itérativement puisque les formes voisines faisant partie du même ensemble que $\mathcal{F}^{(\mathcal{G}_i)}$ influencent la valeur de similarité finale entre $\mathcal{F}^{(r)}$ et $\mathcal{F}^{(\mathcal{G}_i)}$. Pour ce faire, les auteurs se sont inspirés du référencement de pages web (pageRank) développé par la fondation Google Search. Pour calculer le pageRank d'une page web, Google utilise une mesure tenant compte du nombre de pages web, associée à leur pageRank, qui référencent cette page web. De cette manière, plus une page est référencée sur des sites ayant un pageRank élevé, plus cette page aura un pageRank élevé. La Figure 2.30 illustre ce concept. Dans le cas de la similarité entre deux formes, plus une forme requête est fortement similaire à un nombre élevé de formes faisant partie du même ensemble et plus la similarité entre la forme requête et une forme de cet ensemble sera élevée.

D'autres auteurs prennent le parti de créer un arbre de données regroupant les caractéristiques des formes appartenant à la même catégorie (Baseski et al. (2009)) ou sous-catégorie (Erdem et Tari (2010)) pour le comparer à l'arbre créé grâce aux caractéristiques de la forme requête.

Conclusion du chapitre

Dans ce chapitre, nous avons dressé un état de l'art des diverses étapes nécessaires pour la reconnaissance de formes. La première étape réside dans la description de la forme. Pour ce faire, nous avons axé nos travaux sur un descripteur de forme basé squelette car il

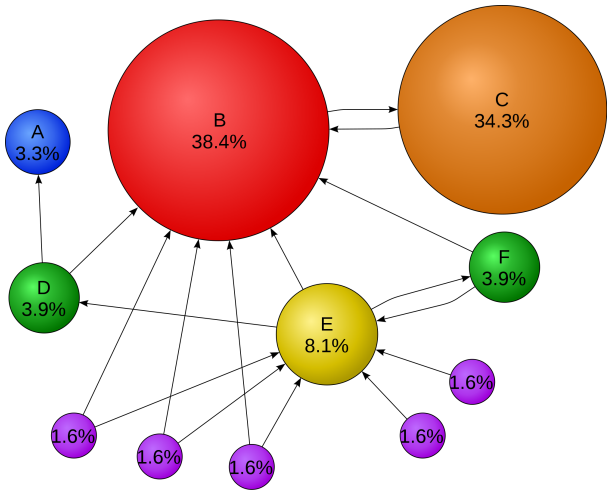


FIGURE 2.30 – *Illustration du calcul du pageRank des pages web. Les sommets du graphe sont les pages web associées à leur pageRank et les arêtes orientées modélisent les hyperliens.*

permet de modéliser la forme globalement et localement. Il existe trois grandes catégories de méthodes permettant d'obtenir le squelette d'une forme. Celles basées sur :

- l'amincissement : elles érodent la forme, en conservant la topologie, jusqu'à ce qu'elle devienne filiforme ;
- le diagramme de Voronoï : elles construisent un squelette continu à partir de l'échantillonnage du bord de la forme ;
- la carte de distance : elles construisent le squelette en déterminant ses crêtes et ses sommets. La construction de notre squelette DECS entre dans cette catégorie de méthodes.

Ces trois types de méthodes ont chacune leurs points forts et leurs points faibles. Leur inconvénient commun est la non-résistance au bruit. Pour éviter ce problème, il est nécessaire de déterminer une hiérarchie entre les branches pour quantifier leur importance. Pour ce faire, les méthodes proposées se basent sur des mesures comme le Rapport de Potentiel de Flexion, la longueur de la partie de la branche qui reconstruit à elle seule une partie de la forme, le pourcentage de reconstruction d'une branche, la simplification du contour, la distance entre deux points du bord en passant par le contour de la forme, les lissages (comme nous le faisons dans notre étape de hiérarchisation) et changements de résolutions successifs. Grâce à cette quantification de l'importance des branches, celles de plus faible signification peuvent être supprimées. Ainsi, un choix binaire à partir de la hiérarchisation doit être fait pour élaguer le squelette. Cela lui permet d'être plus stable au bruit si les chercheurs ne veulent plus utiliser de mesures d'importance dans leurs futures utilisations. Un autre moyen d'ébarbuler le squelette est de lisser la forme avant de le calculer. De cette manière, les branches les moins importantes ne sont pas construites. Le problème est que la forme est modifiée, ce qui entraîne un biais dans la construction du squelette.

Pour apparier les formes à partir de leur squelette, ce dernier peut être modélisé de différentes manières. Certains l'utilisent comme une adjacence de points, d'autres en le transformant en arbre ou encore comme un graphe attribué relationnel. C'est cette dernière solution que nous avons choisie. Quelle que soit la représentation du squelette, le but est d'apparier les

branches entre deux formes pour déterminer une mesure de similarité. En comparant les différentes mesures obtenues, il est possible de déterminer de quelle autre forme, la forme requête est la plus proche et ainsi faire de la reconnaissance ou de la classification de formes.

SQUELETTISATION 3

SOMMAIRE

3.1 EXTRACTION DU SQUELETTE EUCLIDIEN DISCRET CONNECTÉ (DECS) 69
3.1.1 Vue d'ensemble 69
3.1.2 Axe Médian Discret Réduit (RDMA) 70
3.1.3 Filtrage Laplacien de Gaussienne (LoG) 70
3.1.4 Combinaison du RDMA et de la carte des crêtes 73
3.1.5 Représentation par une adjacence de branches 76
3.2 MÉTHODES UTILISÉES POUR COMPARAISON 77
3.2.1 Méthode de Bertrand et Couprie (2014) : Amincissement parallèle basé sur les noyaux critiques 78
3.2.2 Méthode de Choi et al. (2003) : Extraction du squelette Euclidien basé sur un critère de connexité 78
3.2.3 Méthode de Siddiqi et al. (2002) : Squelette Hamilton-Jacobi 79
3.3 RÉSULTATS ET COMPARAISONS 81
3.3.1 Résistance au bruit 81
3.3.2 Influence des paramètres 84
3.3.3 Connexité 84
3.3.4 Complexité 84
3.3.5 Reconstruction 88
3.3.6 Discussion 90
3.4 EXEMPLES DE SQUELETTES 93
CONCLUSION 97

Considérons la reconnaissance d'objets en 2D obtenus après une étape de segmentation d'images. Pour parvenir à résoudre ce problème de reconnaissance, une des méthodes consiste à extraire un ensemble de caractéristiques, dénommé signature, sur la forme à reconnaître (à classifier) et sur les formes représentatives d'une base de données, puis de les comparer.

Le but ici est de représenter la forme grâce à son squelette qui représente une quantité d'informations plus restreinte que celle de la forme d'origine tout en conservant son apparence globale. En d'autres termes, il s'agit d'une compression. En particulier, il est indispensable de conserver les propriétés topologiques de la forme initiale ainsi que ses propriétés géométriques (ramifications, parties allongées par exemple).

Dorénavant, nous considérons que nous ne travaillons que sur des formes connexes. Pour comparer les squelettes, l'idée est de les convertir en une structure de données permettant de manipuler plus facilement les relations existantes entre les branches. Une telle structure peut être un hypergraphe ou un graphe par exemple.

Néanmoins, pour convertir facilement le squelette en une structure de ce type, il est nécessaire qu'il ait les propriétés suivantes :

- Connexité : si le squelette n'est pas connecté, il sera difficile de connecter la structure le représentant. Par conséquent, la topologie du squelette ne serait pas la même que celle de la forme.
- Minceur : les branches doivent avoir un pixel d'épaisseur. Un squelette épais génère des problèmes d'extraction de chemins.

De plus, dans le but d'obtenir des appariements efficaces et pertinents dans un contexte d'objets réels, il est nécessaire de construire des squelettes résistants au bruit. Notons que cette dernière propriété est rarement satisfaite par les algorithmes de la littérature, pour lesquels une légère déformation du bord de la forme génère habituellement une branche (Leyton 1987).

L'algorithme que nous proposons permet d'obtenir un Squelette Euclidien Discret Connecté (ou DECS, pour *Digital Euclidean Connected Skeleton*) en calculant les centres des boules maximales mais exploite également la carte de distance pour les connecter. La contribution majeure de ce travail est l'exploitation conjointe des centres des boules maximales et des crêtes de la carte de distance. Le squelette obtenu est alors connexe, mince et robuste au bruit, comme nous le montrons dans la Partie 5.2.

Avant de décrire la méthode proposée, dans la Partie 3.1, nous détaillons trois méthodes importantes de la littérature grâce à leurs propriétés : amincissement parallèle basé sur les noyaux critiques, appelée "méthode de Bertrand et Couprie (2014)", extraction du squelette Euclidien basé sur un critère de connexité, nommée "méthode de Choi et al. (2003)" et le squelette Hamilton-Jacobi, appelée "méthode de Siddiqi et al. (2002)". Nous avons comparé des méthodes à la nôtre dans la partie 5.2.

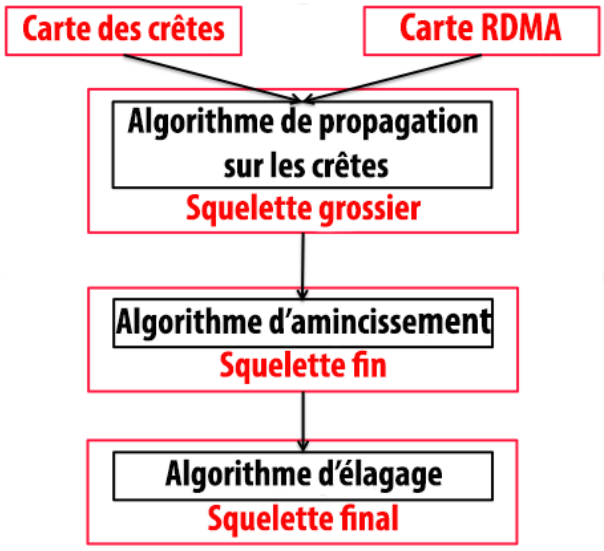


FIGURE 3.1 – *Diagramme de la méthode DECS*

3.1 Extraction du Squelette Euclidien Discret Connecté (DECS)

3.1.1 Vue d'ensemble

Décrivons maintenant, dans le détail, la méthode proposée.

Désignons $\mathcal{I} \subset \mathbb{Z}^2$ une image de taille $M \times N$ et $\mathcal{F} \subset \mathcal{I}$ une forme telle que nous l'avons définie dans la Partie 2 du Chapitre 1.

Selon cette définition, notre squelette sera connexe. Dans le cas où plusieurs formes se trouvent dans $\mathcal{I}$, un squelette peut être extrait sur chaque forme. De plus, il est à noter que $\mathcal{F}$ peut contenir des trous ou pas.

Pour aider à la compréhension, la Figure 3.1 résume la méthode proposée.

La première étape consiste à calculer la carte de distance euclidienne au carré. Puis, parallèlement, nous trouvons le centre des boules maximales et calculons la carte de l'axe médian discret réduit ($RDMA$). La méthode proposée est basée sur la Transformée en Distance Euclidienne au Carré appelée $SEDT$ (pour Squared Euclidean Distance Transformation) que nous décrivons dans la Section 1.5.1 du Chapitre 1 et calculons en utilisant les travaux de Meijster et al. (2002). La carte obtenue contient l'ensemble des centres des boules maximales (*cf.* Section 3.1.2) d'après les travaux de Coeurjolly et Montanvert (2007). Il s'agit des points, pas nécessairement connectés, appartenant au squelette. Notons qu'un tel ensemble est alors appelé axe médian discret. Ensuite, nous utilisons un filtre Laplacien de Gaussienne (filtre LoG) sur la carte de distance euclidienne (*cf.* Section 3.1.3). Puis, l'idée principale est de combiner ces deux cartes dans le but d'obtenir un squelette connecté. En d'autres termes, nous connectons les centres des boules maximales en utilisant les lignes de crête de la carte de distance. Ensuite, nous affinons le squelette obtenu grâce à un post-traitement qui consiste à l'amincir (*cf.* Section 3.1.4) et sélectionnons seulement les branches connectées principales (élagage).

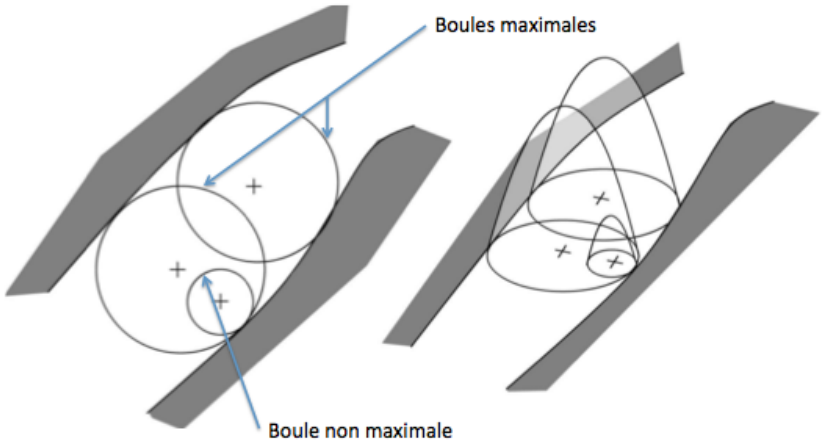


FIGURE 3.2 – *Représentation des boules maximales par des paraboloïdes elliptiques.*

3.1.2 Axe Médian Discret Réduit (RDMA)

À chaque point $\boldsymbol{p}$ de $\mathcal{F}$, la *SEDT* est le carré du rayon de la plus grande boule centrée en $\boldsymbol{p}$. Comme le squelette contient le centre des boules maximales, l'idée principale est de récupérer ces points en utilisant la *SEDT*. Pour cela, il est nécessaire d'avoir un test d'inclusion qui détermine si une boule est recouverte par une autre ou pas.

Considérons la notion de boule maximale discrète donnée dans la définition 1.17 du Chapitre 1. L'ensemble des centres des boules maximales discrètes de $\mathcal{F}$ est l'Axe Médian Discret Réduit (*RDMA*, pour *Reduced Discrete Medial Axis*). Pour obtenir le *RDMA* de $\mathcal{F}$, nous avons utilisé la méthode de Coeurjolly et Montanvert (2007), qui est séparable et a une complexité en $O(n)$, où n est le nombre de pixels dans l'image. En effet, *RDMA* est extrait en seulement quatre passes sur l'image. De plus, cette méthode utilise des valeurs exactes sans approximation. Pour ce faire, les auteurs représentent les boules discrètes par des paraboloïdes elliptiques (comme le montre la Figure 3.2), pour ne conserver que ceux appartenant à l'enveloppe supérieure. Pour illustrer ce concept, supposons que l'on pose un drap, épousant parfaitement les courbes, sur l'ensemble des paraboloïdes elliptiques obtenues pour ne conserver que celles en contact avec ce drap. Les centres des boules maximales sont alors les centres des paraboloïdes elliptiques qui ont été conservés. Un exemple est présenté sur la Figure 3.3

La Figure 3.4 est utilisée pour mettre en évidence le fait que *RDMA* n'est pas nécessairement connexe, ce qui est son inconvénient majeur lorsque nous voulons l'utiliser pour faire de l'appariement de formes. Dans la suite, nous extrayons des caractéristiques sur la carte de distance qui seront utilisées pour construire un squelette connexe.

3.1.3 Filtrage Laplacien de Gaussienne (LoG)

Notons que les branches du squelette correspondent aux crêtes de la carte de distance. L'opérateur Laplacien, qui détermine les variations locales de second ordre, permet de les extraire. Lorsqu'il est appliqué sur la transformée en distance Euclidienne (*EDT*), il est très négatif sur les crêtes positives et proche de zéro sur les pentes linéaires. Nous pouvons donc

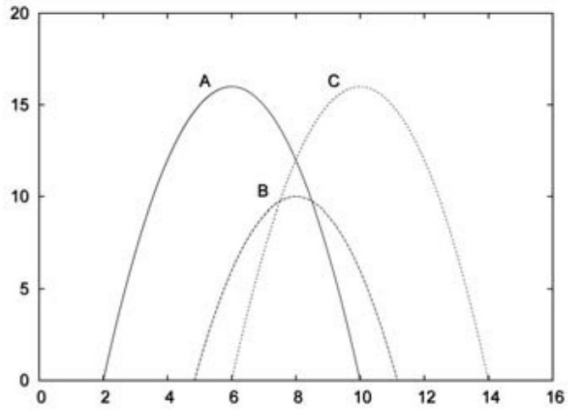


FIGURE 3.3 – *Coupe verticale représentant A,B,C, trois boules maximales. A,C appartiennent au squelette mais pas B car cette dernière est recouverte par l'union de A et de C, d'après Coeurjolly et Montanvert (2007)*

.

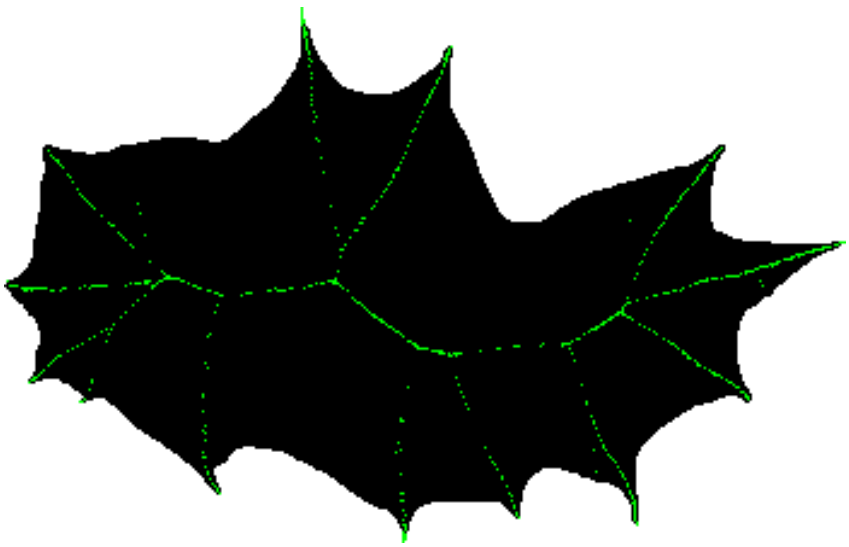


FIGURE 3.4 – *RDMA d'une feuille de houx.*

FIGURE 3.5 – *Carte des crêtes d'une feuille de houx résultant de la convolution de l'EDT avec un filtre LoG négatif.*

considérer le Laplacien négatif de la carte de distance comme une mesure de l'importance des crêtes. Cependant, il est très sensible au bruit lorsqu'il est calculé seul sur la carte de distance. Par conséquent, nous convoluons la carte de distance euclidienne avec le Laplacien négatif d'une gaussienne (*LoG*).

$$RDG(x,y) = -(EDT * LoG)(x,y)$$

avec

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

où l'écart-type σ contrôle l'échelle à laquelle les crêtes sont extraites. Nous avons choisi $\sigma = 1$. Un exemple est montré sur la Figure 3.5. Nous pouvons voir que les branches concernées sont mises en évidence. En pratique, le filtre *LoG* est tronqué au-delà de 3 fois l'écart type. Comme le masque obtenu est séparable, la complexité de l'opération de filtrage est linéaire en σ.

Néanmoins, un simple seuillage de *RDG* est insuffisant pour extraire le squelette car les branches principales, qui doivent être conservées, peuvent se déconnecter, car les valeurs de crête ne sont pas constantes le long des branches. L'idée proposée est de combiner le *RDMA* et la carte des crêtes afin de déterminer l'emplacement des branches concernées (où il y a suffisamment de boules maximales et où la carte des crêtes a des valeurs suffisamment élevées). Cela revient à connecter les centres des boules maximales en utilisant la carte des crêtes qui sert de guide. Nous introduisons deux seuils : $th_{ridge-low}$ correspondant à la valeur minimale de crête détectée et $th_{ridge-high}$ qui est la valeur à partir de laquelle nous considérons une crête comme importante.

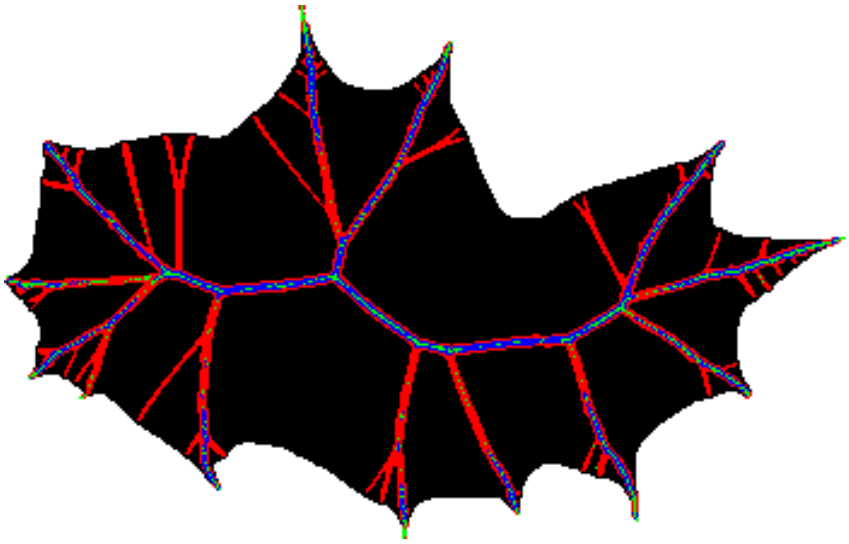


FIGURE 3.6 – *Un exemple de résultat obtenu grâce à l'Algorithme de propagation. Les centres des boules maximales apparaissent en vert, les valeurs de la carte des crêtes supérieures ou égales à $th_{ridge-high}$ sont visibles en bleu et les valeurs de la carte des crêtes comprises entre $th_{ridge-low}$ et $th_{ridge-high}$ sont en rouge.*

3.1.4 Combinaison du RDMA et de la carte des crêtes

Dans cette partie, nous présentons la combinaison de la carte *RDMA* et de la carte des crêtes permettant la construction d'un squelette ayant de bonnes propriétés pour faire de l'appariement basé squelette. Le résumé de toutes les étapes apparaît sur la Figure 3.1.

Donnons d'abord le pseudo-code de l'algorithme de propagation des boules maximales sur les crêtes qui est au cœur de notre proposition (Algorithme 6). Cet algorithme génère un étiquetage

$$h : \mathcal{F} \longrightarrow \{NONE, MAX\_BALL, STRONG\_RIDGE, RIDGE\},$$

où MAX_BALL est l'étiquette des centres des boules maximales, $STRONG\_RIDGE$ est l'étiquette des points ayant une valeur dans la carte des crêtes supérieure ou égale à $th_{ridge-high}$, $RIDGE$ est l'étiquette des points ayant une valeur dans la carte des crêtes entre $th_{ridge-low}$ et $th_{ridge-high}$, et $NONE$ sont les points n'appartenant pas aux branches principales du futur squelette. Les valeurs de paramètres utilisées sont données dans la Section 5.2. L'algorithme utilise une technique de propagation à partir du centre de la plus grande boule maximale. Nous considérons cela comme un point de départ pertinent puisqu'il appartient nécessairement au squelette.

L'algorithme de propagation conserve uniquement les points connectés par un chemin 8-connexe de points de crêtes ou centres de boules maximales. Un exemple de résultat est montré sur la Figure 3.6.

Pour aider à comprendre cet algorithme, nous le détaillons étape par étape. Les lignes 7 à 9 correspondent à l'initialisation de tous les pixels par l'étiquette $NONE$. Dans les lignes 10 à 13, d'abord, nous étiquetons par MAX_BALL le centre de la boule maximale qui a le plus grand rayon. Ensuite, nous l'ajoutons à la pile afin de traiter ses voisins puis, à l'ensemble des

Algorithme 6: Pseudo-code de l'algorithme de propagation sur les crêtes.

Entree : $RDMA$, RDG, $th_{ridge-high}$, $th_{ridge-low}$
Sortie : $h : \mathcal{F} \longrightarrow \{NONE, MAX\_BALL, STRONG\_RIDGE, RIDGE\}$
Variables : $\boldsymbol{p}$,$\boldsymbol{q}$, $max\_SEDT \in \mathbb{Z}^2$,
st une pile de points,
$visited$ un ensemble de points.
début
  pour tous les $\boldsymbol{p} \in \mathcal{I}$ **faire**
    $h(\boldsymbol{p}) \leftarrow NONE$
  fin
  $max\_SEDT \leftarrow \underset{\boldsymbol{p} \in RDMA}{\text{argmax}}\ sedt(\boldsymbol{p})$
  $h(max\_SEDT) \leftarrow MAX\_BALL$
  $ajouter(st, max\_SEDT)$
  $visited \leftarrow \{max\_SEDT\}$
  tant que $nonVide(st)$ **faire**
    $\boldsymbol{p} \leftarrow popTopElement(st)$
    pour tous les $\boldsymbol{q} \in N_8(\boldsymbol{p})$ *tel que* $\boldsymbol{q} \notin visited$ **faire**
      si $\boldsymbol{q} \in RDMA$ **alors**
        $h(\boldsymbol{q}) \leftarrow MAX\_BALL$
        $ajouter(st, \boldsymbol{q})$;
      sinon
        si $RDG(\boldsymbol{q}) \geqslant th_{ridge-high}$ **alors**
          $h(\boldsymbol{q}) \leftarrow STRONG\_RIDGE$
          $ajouter(st, \boldsymbol{q})$
        sinon
          si $th_{ridge-low} \leqslant RDG(\boldsymbol{q}) < th_{ridge-high}$ **alors**
            $h(\boldsymbol{q}) \leftarrow RIDGE$
            $ajouter(st, \boldsymbol{q})$
          fin
        fin
      fin
      $visited \leftarrow visited \cup \{\boldsymbol{q}\}$
    fin
  fin
fin

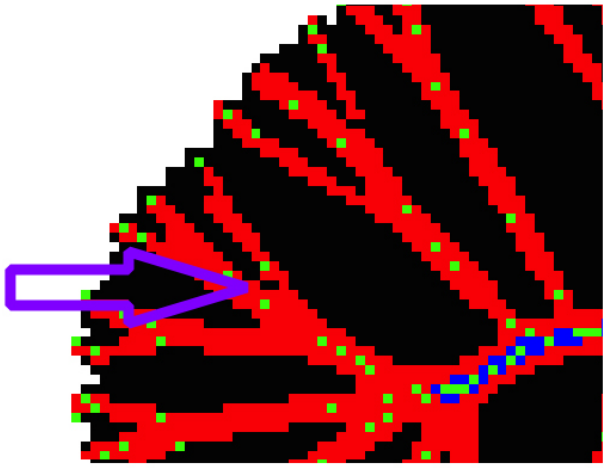


FIGURE 3.7 – *Un exemple de rares trous non désirés.*

pixels visités. Dans les lignes 14-33, nous extrayons le point suivant de la pile. Nous marquons avec *MAX_BALL*, *STRONG_RIDGE* ou *RIDGE* les voisins du pixel courant non visités, si ils correspondent aux critères associés. Ils appartiennent maintenant à l'ensemble des pixels visités et sont ajoutés à la pile. Ces opérations sont répétées jusqu'à ce que la pile soit vide.

À la fin de cet algorithme, le squelette temporaire obtenu, appelé $S_{grossier}$, est l'ensemble des points ayant une étiquette différente de *NONE*. Il contient toutes les branches importantes mais a deux inconvénients. Le premier est que de rares trous indésirables apparaissent dans le squelette à cause du seuillage et de la discrétisation de la carte des crêtes. Ces trous peuvent compromettre le respect de l'homotopie à la forme. Un exemple est donné sur la Figure 3.7.

Pour résoudre ce problème, si les pixels de la composante 4-connexe, dont $\boldsymbol{p}$ fait partie, appartiennent à $\mathcal{F}$ mais n'appartiennent pas à $S_{grossier}$ et qu'aucun n'est 4-connexe à $\bar{\mathcal{F}}$, $\boldsymbol{p}$ et tous les pixels appartenant à sa composante 4-connexe sont ajoutés à $S_{grossier}$ en étant étiquetés *RIDGE*. À la suite de cette étape, l'homotopie de la forme est préservée. Le second inconvénient est que le squelette a une épaisseur qui peut être supérieure à 1.

Pour amincir le squelette, nous avons utilisé l'algorithme d'amincissement MB2 de Manzanera et al. (1999), Bernard et Manzanera (1999). Cet algorithme parallèle permet d'obtenir un ensemble de courbes connectées ayant une épaisseur d'un ou deux pixel(s) (indépendamment du fait que l'épaisseur de la branche correspondante de la forme est paire ou impaire). Pour obtenir une courbe ayant exactement un pixel d'épaisseur, un post-traitement est ajouté. Nous supprimons tous les points simples, qui ne sont pas terminaux grâce au critère de Yokoi. À noter qu'un point est un point terminal s'il n'a qu'un seul voisin dans le squelette. Nous avons choisi cet algorithme linéaire en n en raison de sa complexité. De plus, il est équivalent à un algorithme d'amincissement séquentiel préservant la topologie avec la même règle de suppression Palágyi (2014).

Le résultat de cet algorithme est un squelette fin, appelé S_{fin}, tel que toutes les branches ont une épaisseur d'un pixel au sens de la 8-connexité.

Un exemple de résultat est montré sur la Figure 3.8. À la fin de l'étape courante, le squelette est mince et connecté mais il peut avoir un degré élevé de ramification. La prochaine et

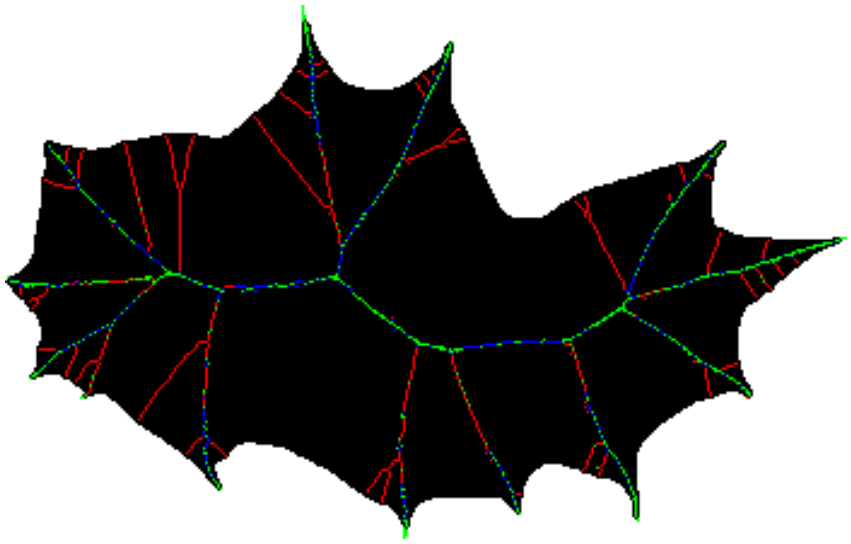


FIGURE 3.8 – *Un exemple de résultat obtenu grâce à l'algorithme d'amincissement MB2 que nous avons modifié pour n'avoir que des branches d'un pixel d'épaisseur.*

dernière étape, qui est facultative, est utilisée pour enlever les branches que nous pouvons qualifier de parasites. En fait, certaines branches ne sont pas nécessaires pour faire de l'appariement de squelettes et peuvent même le perturber.

Nous parcourons la forme et, dès qu'un point terminal est détecté, sa branche est parcourue jusqu'à rencontrer une intersection. Le nombre de points dans la branche est alors stocké ainsi que le nombre de boules maximales et l'endroit où le premier point marqué $STRONG\_RIDGE$ est atteint. Une fois à l'intersection, l'algorithme doit prendre la décision de supprimer la branche ou non. Une branche est supprimée si tous ses points ont une valeur de crête inférieure à $th_{ridge-high}$ et si le pourcentage de boules maximales dans cette branche est inférieur à un seuil $th_{perc-max-ball}$. Dans cet algorithme, chaque point de squelette est visité une seule fois, ce qui donne une complexité linéaire en le nombre de points constituant le squelette.

Finalement, le squelette, appelé S, est constitué de tous les points marqués MAX_BALL, $STRONG\_RIDGE$ ou $RIDGE$. Un exemple de résultat est montré sur la Figure 3.9. Les centres des boules maximales apparaissent en vert, les valeurs de la carte des crêtes supérieures ou égales à $th_{ridge-high}$ sont visibles en bleu et les valeurs de la carte des crêtes comprises entre $th_{ridge-low}$ et $th_{ridge-high}$ sont en rouge. Nous pouvons observer douze exemples de squelette obtenus grâce à la méthode DECS dans la Section 3.4.

3.1.5 Représentation par une adjacence de branches

Comme les branches du squelette ont une épaisseur d'un pixel, le squelette peut être vu comme un ensemble de branches ayant des relations d'adjacence. Une branche est une courbe discrète 8-connectée et, deux branches sont dites adjacentes si elles ont un pixel en commun.

Définissons tout d'abord ce qu'est une branche.

Définition 3.1 (Branche) *Une branche est une séquence* $(\boldsymbol{p}_0, ..., \boldsymbol{p}_{u-1})$ *de u pixels de S tel que pour*

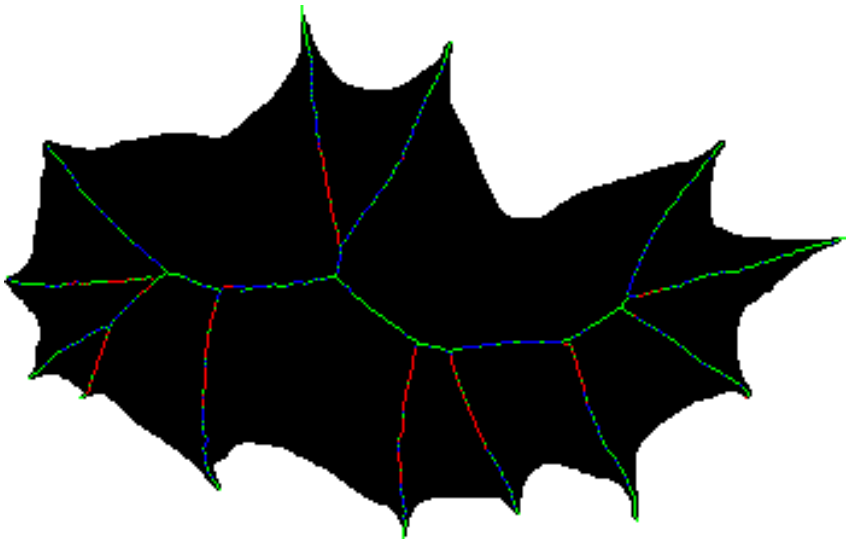


FIGURE 3.9 – *Un exemple de squelette final obtenu avec la méthode DECS après élagage*

tout $i = 1, ...(u-2)$, $N_8(\boldsymbol{p}_i) \cap S = \{\boldsymbol{p}_{i-1}, \boldsymbol{p}_{i+1}\}$ *et pour* $i = 0$ *et* $i = u-1$, $|N_8(\boldsymbol{p}_i) \cap S| \in \{1, 3, 4\}$. *Les extrémités des branches sont* $\boldsymbol{p}_0$ *et* $\boldsymbol{p}_{u-1}$.

Dans un squelette, il existe des branches particulières appelées branches terminales.

Définition 3.2 (Branche terminale) *Une branche* $\mathfrak{b} = (\boldsymbol{p}_0, ..., \boldsymbol{p}_{u-1})$ *est terminale si* $|N_8(\boldsymbol{p}_0) \cap S| = 1$ *ou* $\left|N_8(\boldsymbol{p}_{u-1}) \cap S\right| = 1$, *sinon il s'agit d'une branche interne.*

Définition 3.3 (Ensemble des branches adjacentes en un point) *L'ensemble des branches adjacentes* $\mathcal{B}_A(\mathfrak{b}_i, \boldsymbol{p})$ *à une branche* $\mathfrak{b}_i = (\boldsymbol{p}_0, ..., \boldsymbol{p}_{u-1})$ *en un point* $\boldsymbol{p}$, *où* $\boldsymbol{p} = \boldsymbol{p}_0$ *ou* $\boldsymbol{p}_{u-1}$ *est défini par :*

$$\mathcal{B}_A(\mathfrak{b}_i, \boldsymbol{p}) = \left\{\mathfrak{b}_j = (\boldsymbol{q}_0, ...\boldsymbol{q}_{v-1}) \,\middle|\, (\boldsymbol{q}_0 = \boldsymbol{p} \text{ ou } \boldsymbol{q}_{v-1} = \boldsymbol{p}) \text{ et } \mathfrak{b}_j \neq \mathfrak{b}_i\right\}$$

L'ensemble des définitions précédentes nous amène à considérer le squelette comme un ensemble de branches ayant des relations d'adjacence.

Définition 3.4 (Ensemble des branches adjacentes) *L'ensemble des branches adjacentes* $(\mathcal{B}_A(\mathfrak{b}_i))$ *à une branche* $\mathfrak{b}_i$ *est défini par :*

$$\mathcal{B}^{\backprime}_A(\mathfrak{b}_i = (\boldsymbol{p}_0, ...\boldsymbol{p}_{u-1})) = \mathcal{B}_A(\mathfrak{b}_i, \boldsymbol{p}_0) \cup \mathcal{B}_A(\mathfrak{b}_i, \boldsymbol{p}_{u-1})$$

Définition 3.5 (Chemin de branches) *Un chemin de branches est une séquence* $(\mathfrak{b}_0, ..., \mathfrak{b}_{w-1})$ *de* w *branches telle que pour tout* $i = 0, ...(w-2)$, $\mathfrak{b}_{i+1}$ *appartient à* $\mathcal{B}^{\backprime}_A(\mathfrak{b}_i)$ *et pour tout* $j = 0, ...(w-1)$, $|\mathcal{B}^{\backprime}_A(\mathfrak{b}_j)| \leqslant 2$.

Le squelette S peut donc être vu comme un ensemble de branches dans lequel pour chaque paire de branches $\mathfrak{b}_i$ et $\mathfrak{b}_j$, il existe un chemin de branches entre $\mathfrak{b}_i$ et $\mathfrak{b}_j$.

3.2 MÉTHODES UTILISÉES POUR COMPARAISON

NOUS avons choisi de comparer notre méthode (DECS) à trois méthodes existantes : la méthode de Bertrand et Couprie (2014) est une méthode d'amincissement parallèle récente

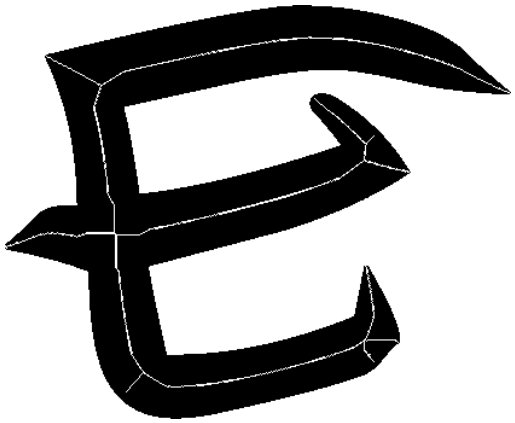


FIGURE 3.10 – *Squelette d'une lettre obtenu grâce à la méthode de Bertrand et Couprie (2014).*

développée simultanément à la nôtre alors que celle de Choi et al. (2003) et celle de Siddiqi et al. (2002) sont basées sur une carte de distance, comme la méthode proposée.

3.2.1 Méthode de Bertrand et Couprie (2014) : Amincissement parallèle basé sur les noyaux critiques

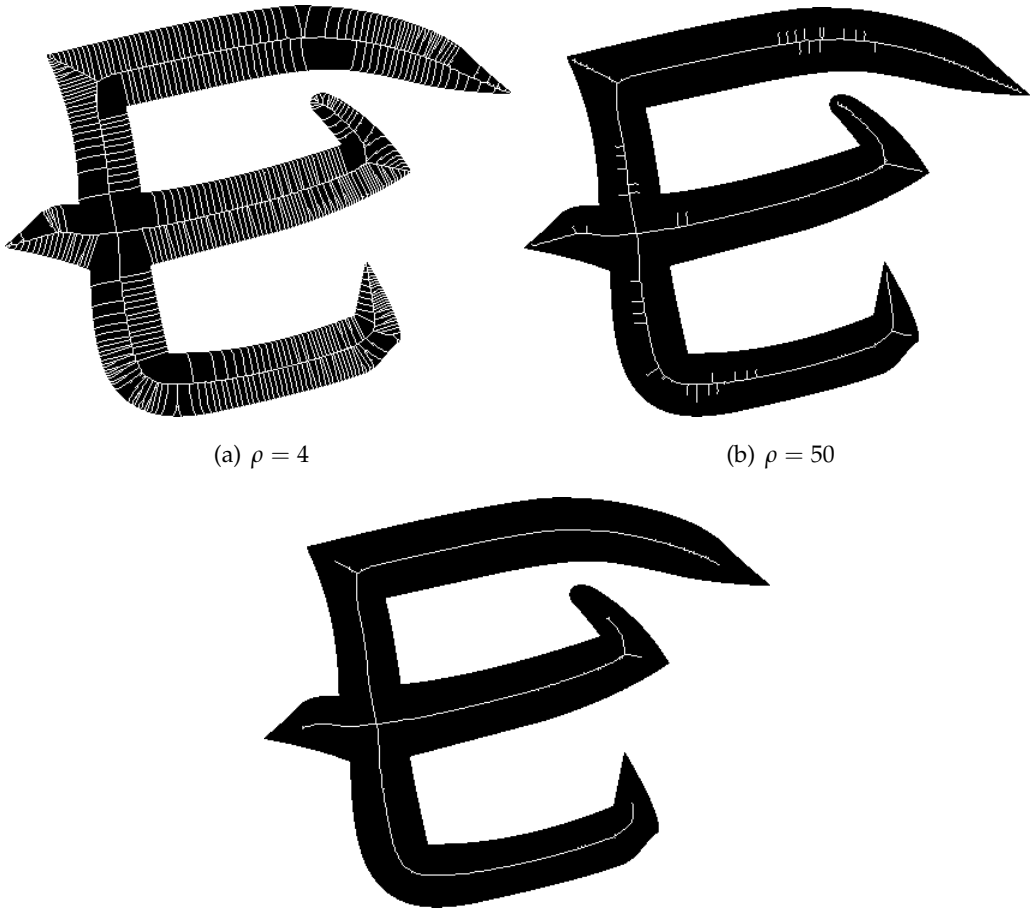
Il s'agit d'une méthode d'amincissement parallèle basée sur les noyaux critiques. L'idée principale est d'amincir progressivement la forme jusqu'à stabilité. Cet algorithme s'appuie sur les complexes abstraits. Intuitivement, ce sont des ensembles d'éléments de dimensions variables (cubes, carrés, segments et points) ayant des règles spécifiques entre chacun de ces éléments. Le principe est de supprimer des points simples parallèlement sans modifier la topologie de la forme. Cette définition s'appuie sur l'opération d'effrondrement : un outil classique en topologie algébrique qui préserve la topologie. Cela découle du fait que, si un sous-ensemble Y de X contient le noyau critique (une rétraction homotopique) de X, alors Y a la même topologie que X. Nous pouvons observer sur la Figure 3.10, un exemple de squelette obtenu grâce à la méthode de Bertrand et Couprie (2014).

Il est à noter que cette méthode n'a pas de paramètres permettant de mettre au point le squelette voulu, comme nombre d'autres méthodes de squelettisation. Cela implique une faible résistance au bruit que nous montrerons dans la Partie 5.2.

3.2.2 Méthode de Choi et al. (2003) : Extraction du squelette Euclidien basé sur un critère de connexité

Cette méthode génère un squelette Euclidien connecté. Cet algorithme commence par le calcul de la carte de Distance Euclidienne Séquentielle Signée 8-connectée (8SSED) construite par Ye (1988) et expliquée dans le Chapitre 1.

L'étape suivante est l'extraction du squelette basée sur un critère de connexité utilisant un seuil ρ. La complexité de cet algorithme est linéaire en n, le nombre de pixels de l'image. Comme illustré sur la Figure 3.11, le nombre de branches du squelette décroît lorsque ρ augmente. Cela permet de trouver la valeur du seuil appropriée en fonction de l'application désirée. Cette méthode est intéressante car elle a été utilisée, en appariement de graphes, par



(a) $\rho = 4$ (b) $\rho = 50$

(c) $\rho = 800$

FIGURE 3.11 – *Squelettes obtenus par la méthode de Choi et al. (2003) en utilisant différents seuils.*

Shen et al. (2013). De plus, il s'agit d'une méthode de squelettisation reposant sur une carte de distance, comme DECS.

3.2.3 Méthode de Siddiqi et al. (2002) : Squelette Hamilton-Jacobi

Tout comme la méthode précédente, celle-ci génère un squelette Euclidien ayant un pixel d'épaisseur. Elle repose sur une modélisation initiale en continu de la distance Euclidienne à la forme d'entrée qui est basée sur le gradient de la carte de distance. Le champ de vecteurs obtenu est fortement orienté vers les crêtes de la carte, comme le montre la figure 3.12. La valeur de crête d'un point est liée à la quantité de vecteurs pointant sur lui.

Cette méthode partage beaucoup de propriétés avec celle proposée étant donné qu'elle s'appuie sur la carte de distance. Elle utilise également un seuil (th_{AOF}). Sur la Figure 3.13, nous pouvons remarquer que les branches les moins importantes disparaissent lorsque th_{AOF} décroît.

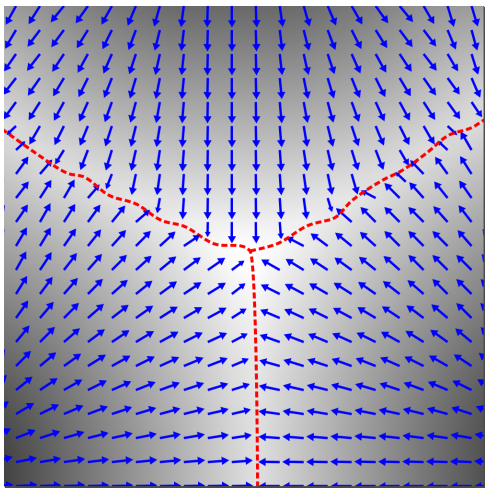
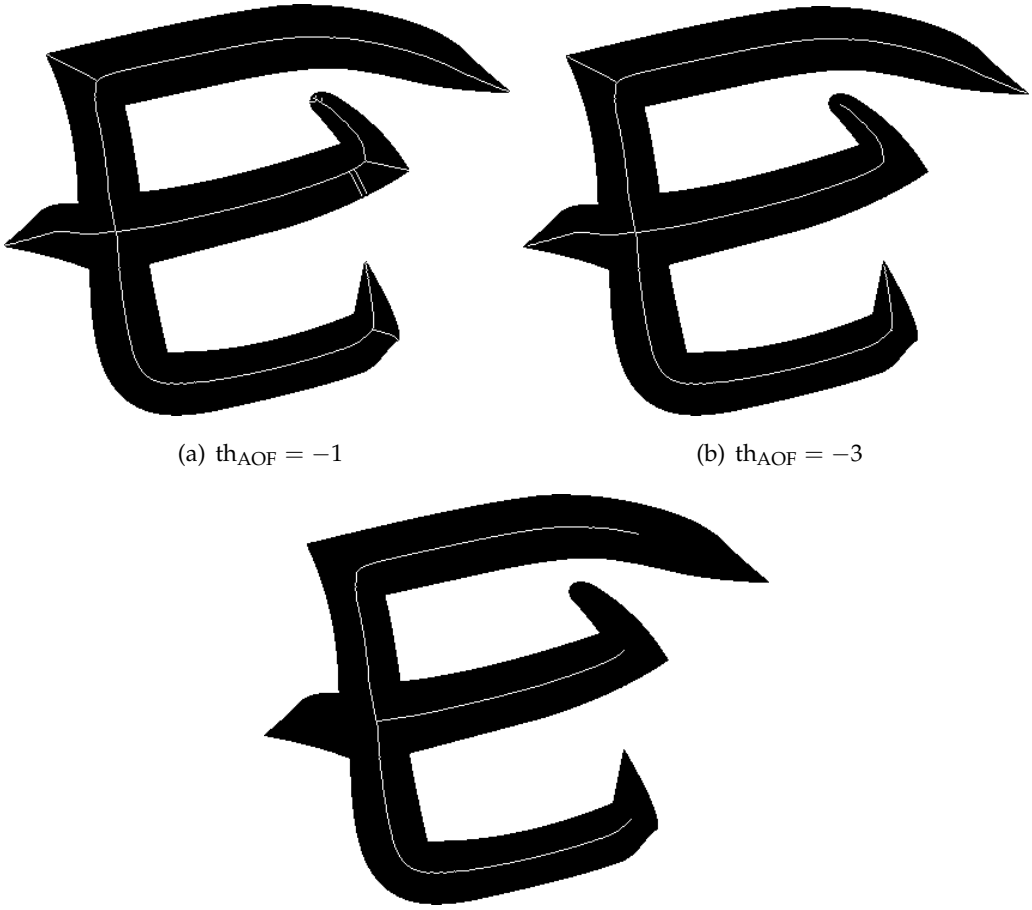


FIGURE 3.12 – *Champ de vecteurs généré par le calcul du gradient de la transformée en distance Euclidienne. Les vecteurs sont dirigés vers les crêtes de la carte de distance. Les points se situant à l'intersection des flux sont suceptibles d'être des points de squelette. Ils sont représentés par les lignes en pointillé rouge.*



(a) $\text{th}_{\text{AOF}} = -1$ (b) $\text{th}_{\text{AOF}} = -3$

(c) $\text{th}_{\text{AOF}} = -5$

FIGURE 3.13 – *Squelettes obtenus grâce à la méthode de Siddiqi et al. (2002) en utilisant différents seuils.*

3.3 RÉSULTATS ET COMPARAISONS

DANS cette section, nous testons diverses propriétés telles que la connexité, la capacité à reconstruire et la tolérance au bruit. Ainsi, nous confrontons notre méthode à celles décrites dans la Partie 3.2 en fonction des critères énoncés précédemment. Nous faisons également une comparaison de leur complexité. Tous les tests sont effectués sur une base de données de formes, créée par Latecki et al. (2000), ayant environ un millier de formes.

3.3.1 Résistance au bruit

Pour tester la tolérance au bruit, nous avons créé 15 squelettes théoriques : manuellement, nous avons dessiné des squelettes 8-connexes ayant un pixel d'épaisseur. En parcourant automatiquement chaque branche $(\boldsymbol{p}_1, ...\boldsymbol{p}_i, ...\boldsymbol{p}_u)$, nous avons attribué des rayons de boules maximales aux points des branches, en utilisant des fonctions linéaires, sinusoïdales ou logarithmiques en fonction de i. Nous avons pris soin de générer des courbes pour lesquelles le rayon de courbure était assez élevé en tous points (pas de concavités ni de convexités élevées). Ensuite, nous avons reconstruit des formes en utilisant les rayons de boules maximales pour obtenir des formes théoriques. Du bruit a été ajouté en déplaçant aléatoirement chaque pixel de contour vers l'intérieur ou vers l'extérieur de la forme dans la direction normale au contour. Le déplacement a été tiré aléatoirement selon une loi uniforme sur l'intervalle $[-k, k]$. Nous nous sommes assurés que l'épaisseur de la forme était suffisamment élevée pour ne pas en rompre la connexité.

Lorsque tous les pixels de la frontière ont été déplacés, nous avons calculé une courbe fermée 8-connexe à partir de la liste des pixels déplacés et ensuite nous avons recréé une forme en coloriant les pixels se situant à l'intérieur de cette courbe pour ensuite recalculer un squelette. Pour nos tests, nous avons utilisé $bruit_1$ avec $k = 1$ et $bruit_2$ avec $k = 3$. Pour chaque méthode, nous avons sélectionné un squelette de référence. Pour cela, nous avons fait varier le (ou les) seuil(s) de la méthode et généré un squelette pour chaque valeur de seuil(s). Nous avons conservé le squelette ayant la plus petite Distance de Hausdorff Modifiée (MHD) par rapport au squelette théorique.

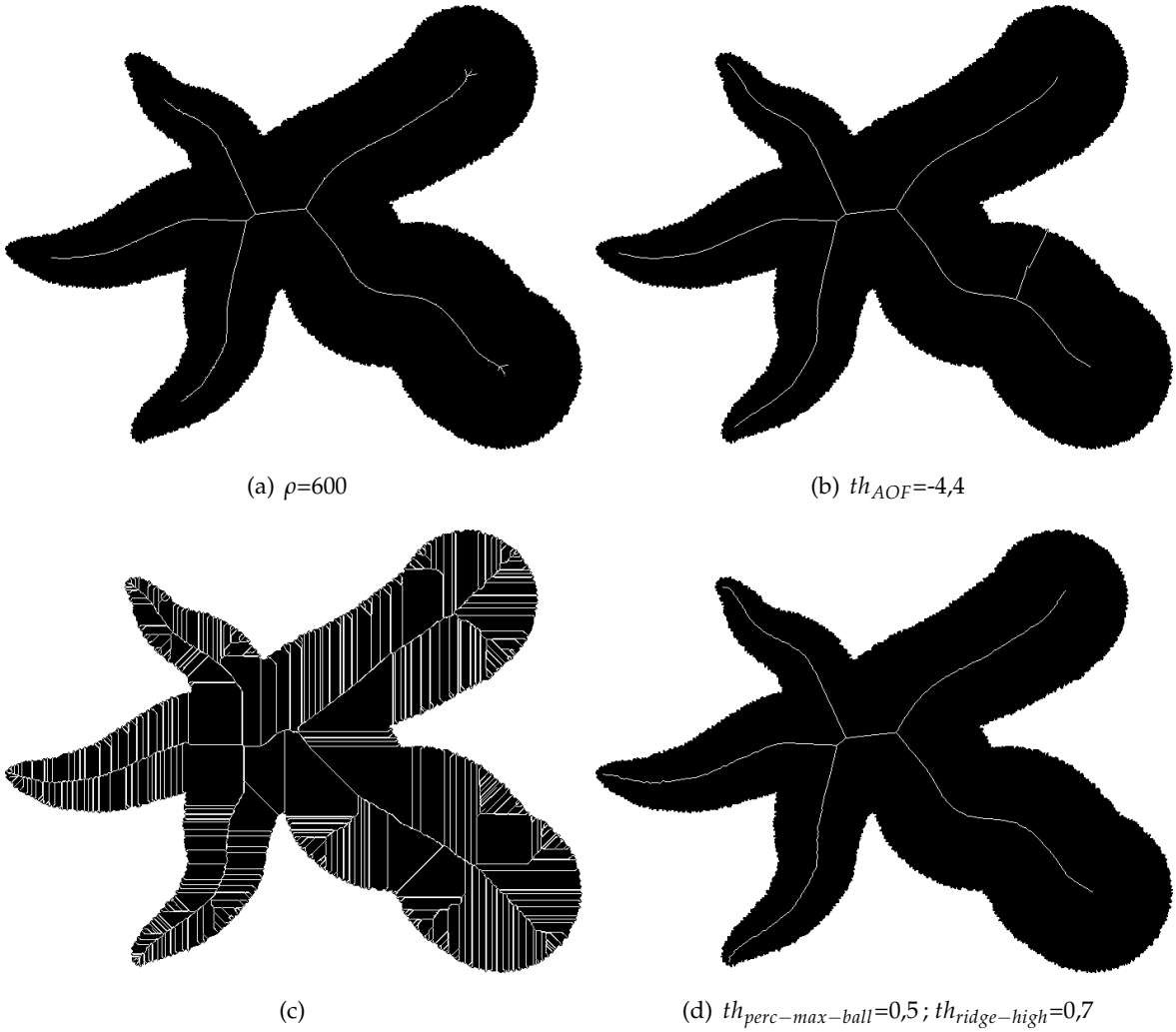
Soit P, l'ensemble des pixels du squelette théorique et Q, l'ensemble des pixels du squelette généré (P et Q sont deux sous-ensembles non-vides de $\mathbb{Z}^2$).

Définition 3.6 *La **Distance de Hausdorff Modifiée** $MHD(P, Q)$ est définie par :*

$$MHD(P,Q) = \max\left\{ \frac{1}{|P|} \sum_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q}\{d_E(\boldsymbol{p},\boldsymbol{q})\}, \frac{1}{|Q|} \sum_{\boldsymbol{q}\in Q} \min_{\boldsymbol{p}\in P}\{d_E(\boldsymbol{q},\boldsymbol{p})\} \right\},$$

où d_E est la distance Euclidienne.

En d'autres termes, la distance de Hausdorff modifiée est la distance moyenne entre le squelette théorique et le squelette généré. Pour chaque bruit, méthode et image, nous calculons la plus petite MHD entre le squelette retenu et celui de référence. La Figure 3.14 fournit un exemple de squelettisation de forme bruitée pour chaque méthode avec des seuils adaptés



(a) ρ=600 (b) th_{AOF}=-4,4

(c) (d) $th_{perc-max-ball}$=0,5 ; $th_{ridge-high}$=0,7

FIGURE 3.14 – *Résultat de la squelettisation de forme bruitée pour chaque méthode avec le seuil le plus adapté (lorsqu'il existe), c'est-à-dire donnant la plus petite MHD, a) Méthode de Choi et al. (2003), b) Méthode de Siddiqi et al. (2002), c) Méthode de Bertrand et Couprie (2014) and d) Méthode DECS (Leborgne et al. 2015).*

(lorsque ceux-ci existent), c'est-à-dire donnant la plus petite MHD. On peut noter que la méthode de Bertrand et Couprie (2014) ne résiste pas au bruit. Nous n'avons donc pas quantifié, par la suite, la tolérance au bruit pour cette méthode.

Les résultats de cette expérience sont présentés sur la Figure 3.15. Ce graphique montre que la méthode de Siddiqi et al. (2002) n'est pas résistante au bruit au regard des autres méthodes. De plus, cela démontre que la méthode de Choi et al. (2003) et la nôtre sont très proches en dépit d'un léger avantage à peine perceptible à l'œil nu sur les images pour la méthode de Choi et al. (2003). Néanmoins, le problème majeur de leur méthode réside dans le fait que le squelette n'est pas toujours connexe comme nous l'expliquons dans la Partie 3.3.3. La méthode de Choi et al. (2003) n'est donc pas utilisable dans notre cas.

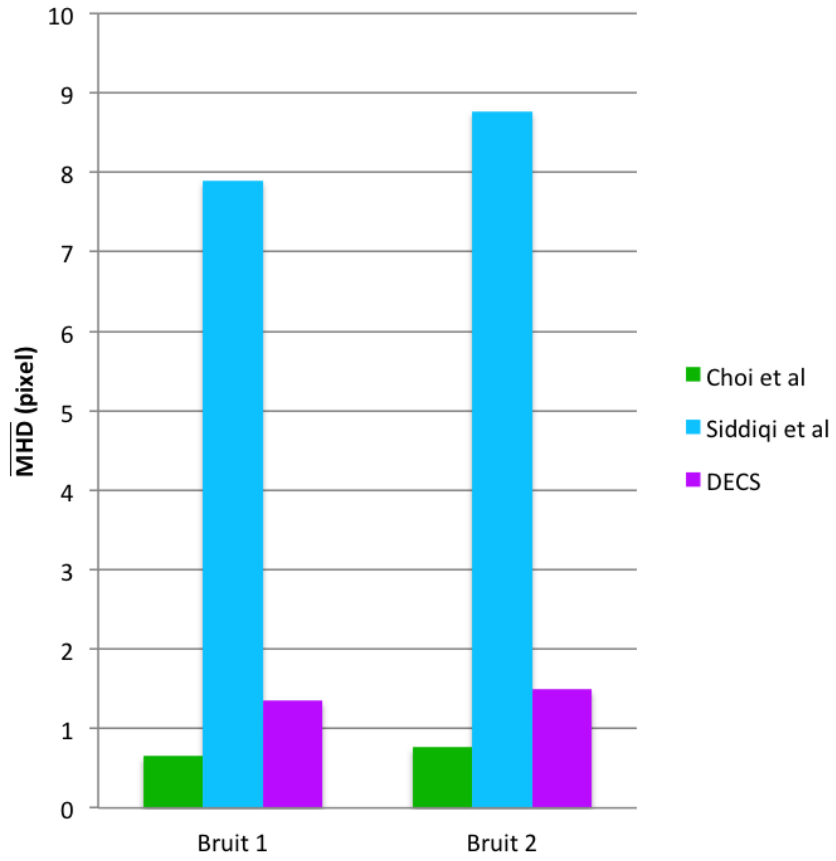


FIGURE 3.15 – *Comparaison de la résistance au bruit. Pour i : indice du squelette* $\in [1 \dots 15]$*, pour t comme* $th_{AOF} \in [-6 \dots -0,1]$ *pour la méthode de Siddiqi et al. (2002), pour t comme* $\rho \in [4 \dots 3000]$ *pour la méthode de Choi et al. (2003) et t est la combinaison de* $th_{perc-max-ball} \in [0 \dots 1]$ *et* $th_{ridge-high} \in [0,1 \dots 1,1]$ *pour la méthode DECS. La MHD représentée est une moyenne des plus petites MHD obtenues sur chaque image.*

3.3.2 Influence des paramètres

Dans la Partie 3.3.1, nous avons présenté des tests sur des images réelles. Chaque méthode a été testée dans les meilleures conditions : pour chaque image, nous avons choisi le(s) seuil(s) qui permettai(en)t d'obtenir les meilleurs résultats au regard de notre critère d'évaluation. Pour la méthode de Choi et al. (2003), la quantité de branches parasites diminue au fur et à mesure que ρ augmente. La moyenne des seuils qui permettent d'obtenir les meilleurs résultats est $\rho = 803$. Pour la méthode de Siddiqi et al. (2002), la quantité de branches parasites diminue lorsque le seuil th_{AOF} diminue. La moyenne des seuils donnant les meilleurs résultats est égale à -3.06. En ce qui concerne la méthode proposée, la suppression de branches non désirées dépend de la combinaison de deux seuils. Ceux-ci varient dans le même sens, à savoir que plus $th_{perc-max-ball}$ et $th_{ridge-high}$ sont élevés, moins il y a de branches parasites.

$th_{ridge-low}$ est toujours égal 0.05 et $th_{ridge-high}$ varie entre $th_{ridge-low}$ et 1.1. Les seuils donnant les meilleurs résultats pour cette méthode sont $th_{perc-max-ball} = 0.4$ et $th_{ridge-high} = 0.5$. Cependant, la Figure 3.16 montre quatre squelettes obtenus grâce à la méthode DECS pour une même forme mais avec des seuils différents pour donner une idée générale de l'influence des paramètres.

3.3.3 Connexité

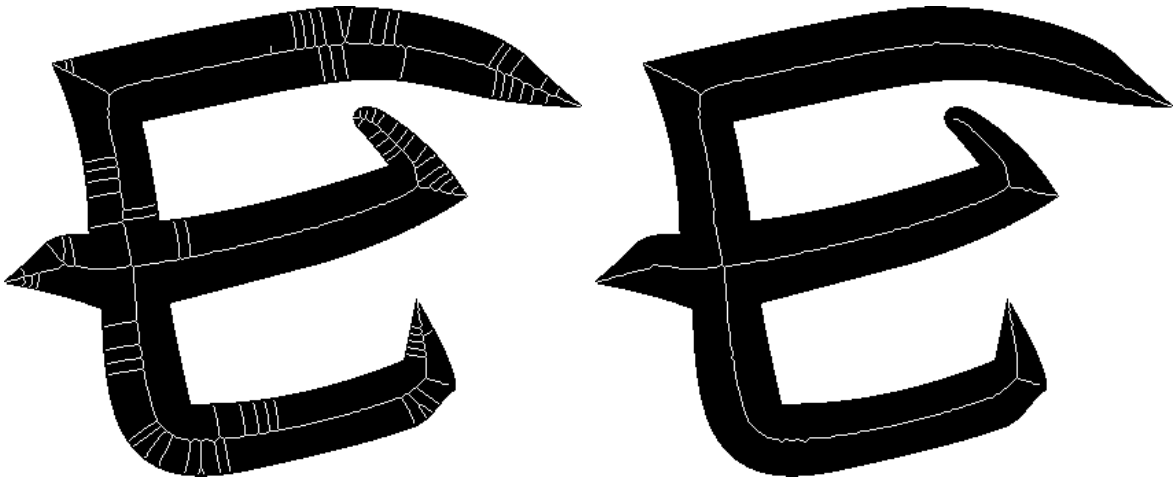
Les méthodes de Bertrand et Couprie (2014), Siddiqi et al. (2002) et DECS permettent de construire un squelette connecté dans n'importe quelle situation. En revanche, pour la méthode de Choi et al. (2003), lorsque le seuil devient élevé, la connexité du squelette n'est pas garantie. Quelques déconnexions sont mises en évidence sur la Figure 3.17. Sur l'exemple, le squelette se déconnecte à de nombreux endroits sur le bord de la forme même si le seuil est relativement petit. Des déconnexions sont également présentes lorsque l'épaisseur de la forme diminue et forme un choc de deuxième ordre (un goulet) d'après Siddiqi et al. (1999). Par conséquent, la topologie de la forme est perdue, ce qui est un inconvénient majeur lors de l'appariement de squelettes.

Pour ce critère, cette dernière méthode n'est pas intéressante car elle ne respecte pas la topologie de la forme. Notons que la conservation de la topologie est primordiale pour nous puisque notre but est de calculer un squelette devant être homotope à la forme afin de le représenter par un hypergraphe qui, lui aussi, doit être homotope à la forme.

3.3.4 Complexité

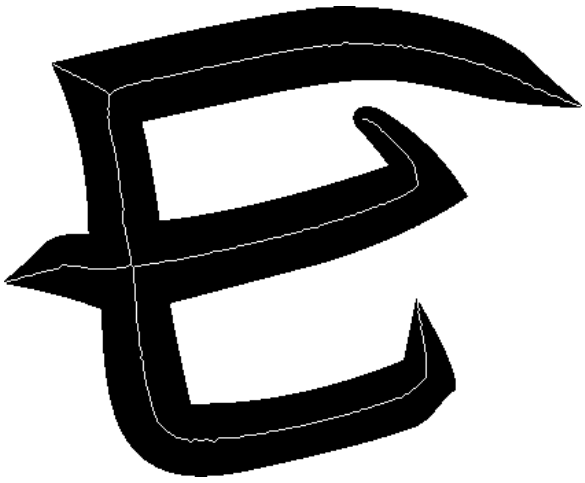
Dans ce paragraphe, nous rappelons que n est le nombre de pixels de l'image. La méthode proposée a une complexité linéaire. Une étude expérimentale a été réalisée sur toutes les formes de la base de données de Latecki et al. (2000) et est représentée sur la figure 3.18(a).

Pour prouver cela, nous détaillons la complexité de chaque étape. Le calcul de la carte *SEDT* est linéaire, comme le prouvent Coeurjolly et Montanvert (2007) mais également Meijster et al. (2002). Le calcul de la carte des crêtes est également linéaire car il s'agit d'une convo-

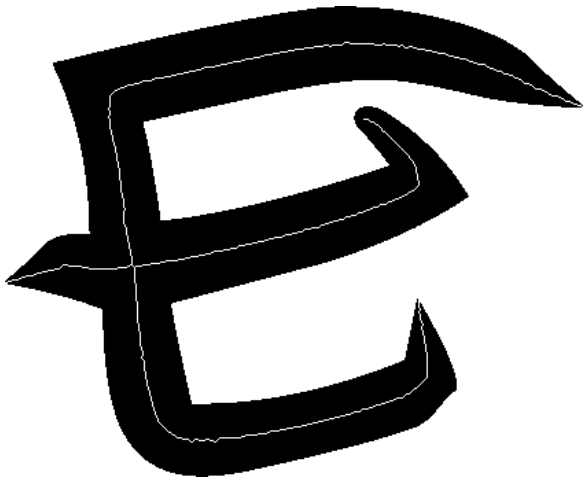


(a) $th_{perc-max-ball}$=0 ; $th_{ridge-high}$=0.1

(b) $th_{perc-max-ball}$=0.2 ; $th_{ridge-high}$=0.7

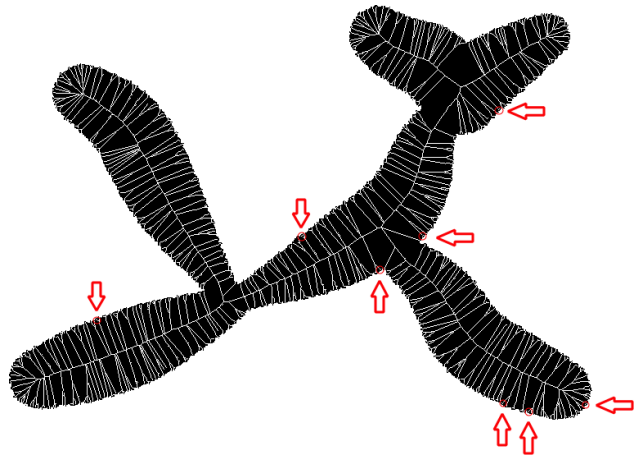


(c) $th_{perc-max-ball}$=0.35 ; $th_{ridge-high}$=1.1

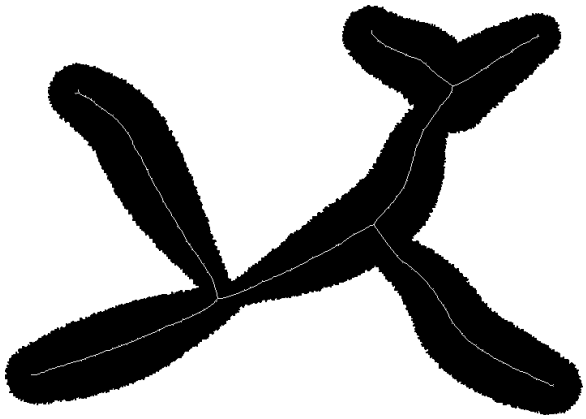


(d) $th_{perc-max-ball}$=0.5 ; $th_{ridge-high}$=0.8

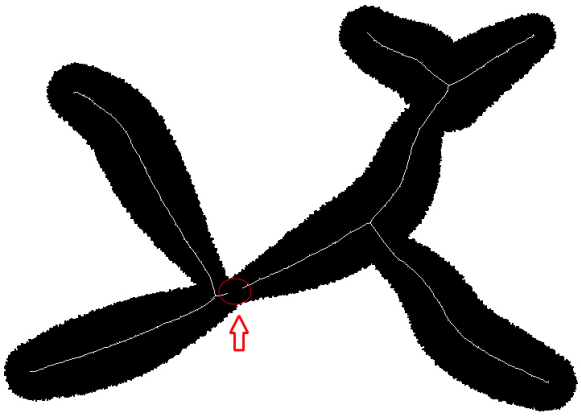
FIGURE 3.16 – *Exemples de squelettes obtenus grâce à la méthode proposée en faisant varier les deux seuils.* $th_{ridge-low}$ *vaut toujours 0.05.*



(a) ρ=9

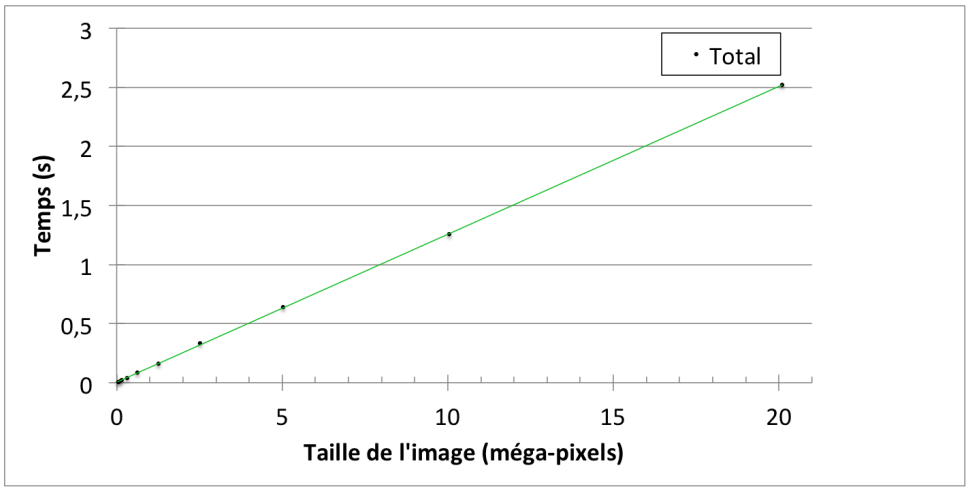


(b) ρ=1100

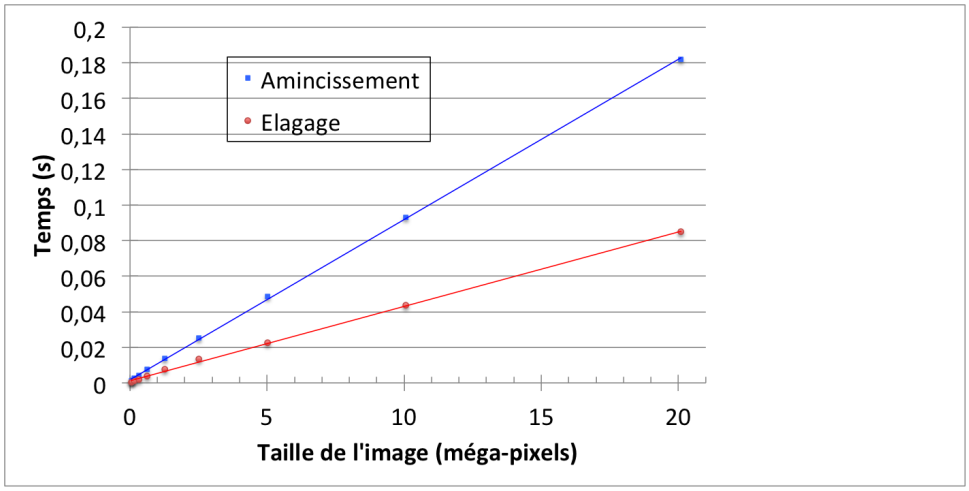


(c) ρ=1200

FIGURE 3.17 – *Évolution du seuil dans le but de mettre en évidence les déconnexions des squelettes obtenus avec la méthode de Choi et al. (2003).*



(a)



(b)

FIGURE 3.18 – *Ces graphiques présentent la moyenne des temps de calcul de squelettes pour chaque image de la base de Latecki et al. (2000) en fonction de n, le nombre de pixels de l'image. Pour faire varier n, entre deux itérations, la hauteur et la largeur sont multipliées par* $\sqrt{2}$*. Par conséquent, n double à chaque itération. a) Temps de calcul total, b) Temps d'élagage (en rouge) et temps d'amincissement (en bleu).*

lution de la carte de distance euclidienne avec le filtre *LoG*. Donc, la complexité est en $O(n\sigma)$. L'extraction des centres de boules maximales est également linéaire car elle se compose de quatre passes linéaires sur l'image (Coeurjolly et Montanvert 2007). En ce qui concerne l'algorithme de propagation, nous choisissons, à partir de la boule maximale ayant le plus grand rayon, les points de squelette. Pour cette étape, nous parcourons tous les pixels de la forme, au plus une fois. Donc, cette étape a également une complexité linéaire. La linéarité de l'amincissement et de l'élagage n'est pas simple à prouver par le raisonnement, mais elle est observée sur le graphique de la Figure 3.18(b).

La méthode de Choi et al. (2003) a une complexité linéaire aussi. De plus, nous pouvons assimiler la complexité de la méthode de Bertrand et Couprie (2014) à une complexité linéaire car il s'agit d'un algorithme d'amincissement parallèle (même si ce n'est pas explicitement écrit dans leur publication). La méthode de Siddiqi et al. (2002) a une complexité en O(n log n). Au regard de ce critère, cette dernière méthode est moins intéressante.

3.3.5 Reconstruction

L'une des propriétés importantes du squelette est sa capacité à reconstruire la forme. Cela montre comment la géométrie de la forme est préservée. En fait, cette propriété garantit qu'une méthode de squelettisation conserve les informations initiales, c'est-à-dire qu'elle ne supprime ni ne génère de pixels. La nécessité d'une reconstruction exacte est discutable étant donné que cela dépend de l'application. En cas d'appariement, il n'est pas nécessaire de reconstruire la forme exactement, mais nous avons besoin d'obtenir approximativement la forme initiale.

L'avantage de la méthode proposée est que, en fonction des seuils fixés, nous obtenons différents niveaux de reconstruction. En fait, une branche de squelette est supprimée si elle ne contient aucun pixel ayant une valeur de crête supérieure ou égale à $th_{ridge-high}$ou si le pourcentage de boules maximales dans cette branche est inférieure à $th_{perc-max-ball}$ (*cf.* Section 3.1.4). Ainsi, le niveau de détails du squelette augmente à mesure que $th_{perc-max-ball}$et $th_{ridge-high}$décroissent.

Un exemple de squelette et son image de différences associée sont présentés sur la Figure 3.19.

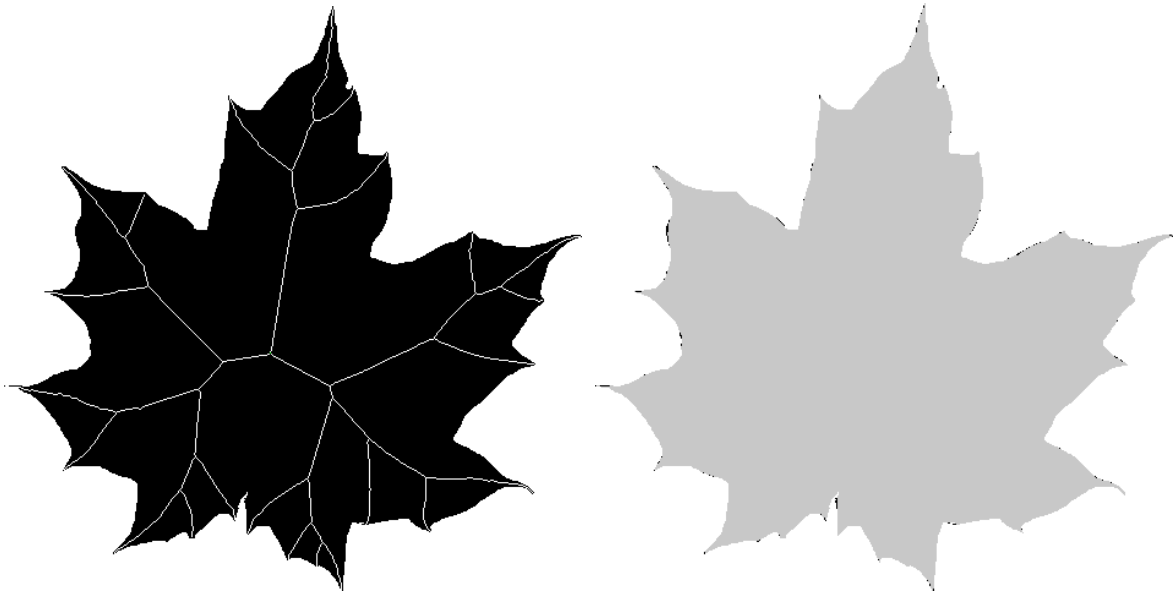
Il convient de noter que, dans le but d'effectuer la reconstruction, la valeur de la *SEDT* est conservée pour chaque point du squelette. Ainsi, il suffit de dessiner une boule ayant le rayon correspondant, centré sur chaque point de squelette. Nous introduisons maintenant trois mesures d'évaluation de la reconstruction.

Soit P, l'ensemble des pixels de la forme originale et Q, l'ensemble des pixels de la forme reconstruite (P et Q sont deux sous-ensembles non-vides de $\mathbb{Z}^2$).

Définition 3.7 *Le **pourcentage de forme non reconstruite** est défini par :*

$$Perc = \frac{|P \setminus Q|}{|P|} \times 100$$

Définition 3.8 *La **Distance de Hausdorff** $HD(P, Q)$ est définie par :*



(a) (b)

FIGURE 3.19 – *(a) Exemple de squelette extrait à partir d'une feuille grâce à la méthode DECS ($th_{perc-max-ball}$=0.4 et $th_{ridge-high}$=0.5), (b) Image de différences obtenue. L'image reconstruite est représentée en gris et les pixels qui appartiennent à la forme d'origine mais pas à la forme reconstruite sont de couleur noire.*

$$HD(P,Q) = \max\left\{ \max_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q} d_E(\boldsymbol{p},\boldsymbol{q}), \max_{\boldsymbol{q}\in Q} \min_{\boldsymbol{p}\in P} d_E(\boldsymbol{q},\boldsymbol{p}) \right\},$$

où d_E est la distance Euclidienne.

En d'autres termes, la distance de Hausdorff est la distance maximale entre la forme d'origine et la forme reconstruite.

Définition 3.9 *La **Distance de Hausdorff Modifiée** $MHD(P,Q)$ est définie, comme précédemment, par :*

$$MHD(P,Q) = \max\left\{ \frac{1}{|P|}\sum_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q}\{d_E(\boldsymbol{p},\boldsymbol{q})\}, \frac{1}{|Q|}\sum_{\boldsymbol{q}\in Q} \min_{\boldsymbol{p}\in P}\{d_E(\boldsymbol{q},\boldsymbol{p})\} \right\},$$

En d'autres termes, la distance de Hausdorff modifiée est la distance moyenne entre la forme originale et la forme reconstruite.

Dans la suite de cette partie, nous avons comparé le squelette généré grâce à la méthode DECS à ceux obtenus à partir des méthodes de la section 3.2. Pour que les comparaisons soient aussi justes que possible, les squelettes obtenus avec les trois méthodes existantes ont été post-traités grâce aux travaux de Bernard et Manzanera (1999) afin d'obtenir des branches d'un pixel d'épaisseur. Les résultats de ce premier test peuvent être retrouvés dans le tableau 3.1 dans lequel les valeurs correspondent à la moyenne des mesures obtenues pour les 1071 formes contenues dans la base de données. Pour chacune des quatre méthodes, nous avons sélectionné les meilleurs résultats de reconstruction en faisant varier les seuils.

TABLE 3.1 – *Quantification des meilleures reconstructions.*

	Perc	HD	MHD
Méthode de Siddiqi et al. (2002)	0.58216	1.20559	1.04608
Méthode de Choi et al. (2003)	0.37558	1.19915	0.95196
Méthode de Bertrand et Couprie (2014)	0.37334	1.17093	1.02878
DECS	0.52732	1.19121	0.99731

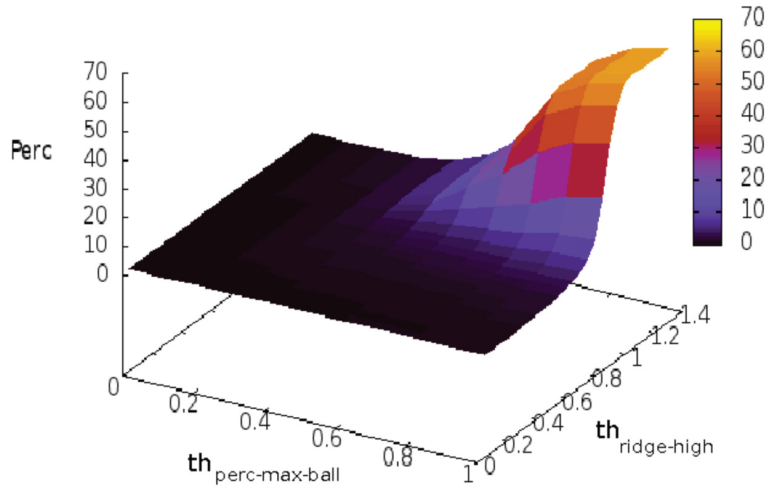
Nous pouvons d'ores et déjà noter que la reconstruction parfaite est impossible étant donné que nous avions choisi d'extraire les squelettes d'une épaisseur d'un pixel. En effet, lorsque l'épaisseur de la forme est paire, afin de pouvoir reconstruire la forme originale précisément, le squelette doit avoir une épaisseur de 2 pixels pour rester centré. Néanmoins, dans une image 600×400, cela fait une différence d'au plus 1 pixel sur le contour reconstruit, ce qui est imperceptible lors d'une inspection visuelle rapide. En ce qui concerne la meilleure reconstruction, nous pouvons considérer que ces quatre méthodes sont comparables.

Les Figures 3.20, 3.21 et 3.22 montrent que le changement des seuils affecte la qualité de la reconstruction aussi bien que la quantité de détails. Ces paramètres sont responsables de la densité de ramification du squelette. Diminuer le degré de ramification améliore la facilité d'utilisation du squelette (dans un processus d'appariement, par exemple), mais il diminue sa capacité à reconstituer avec précision la forme souhaitée. Il s'agit alors de trouver un compromis entre la précision de la reconstruction du squelette non élagué et la facilité d'utilisation du squelette élagué dans un traitement ultérieur. Alors qu'un squelette élagué peut ne pas parfaitement reconstruire la forme, il est souvent préférable de l'utiliser à des fins d'appariement de formes. La première observation que nous pouvons faire est que les trois mesures varient de la même manière en fonction de la variation des seuils. Cependant, la qualité de la reconstruction de la méthode de Choi et al. (2003) se détériore très rapidement tandis que le paramètre de la méthode de Siddiqi et al. (2002) peut varier dans une large gamme de valeurs avant que les résultats ne se détériorent.

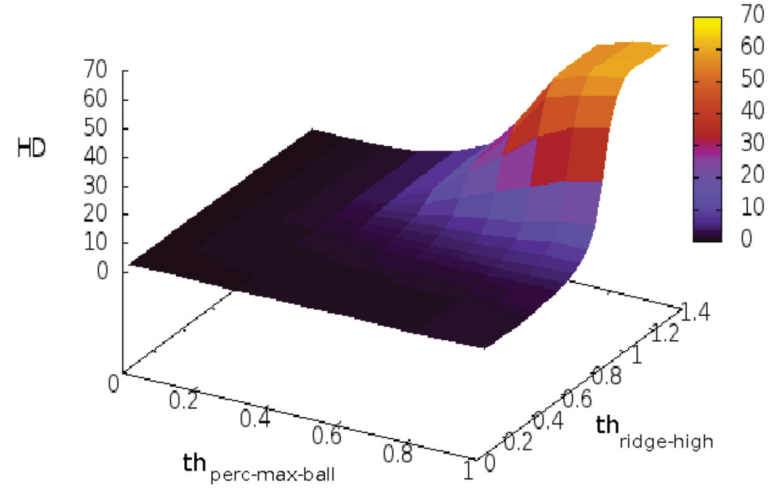
3.3.6 Discussion

Rappelons-nous que les critères de squelette, qui sont les plus critiques pour l'appariement de formes, sont :

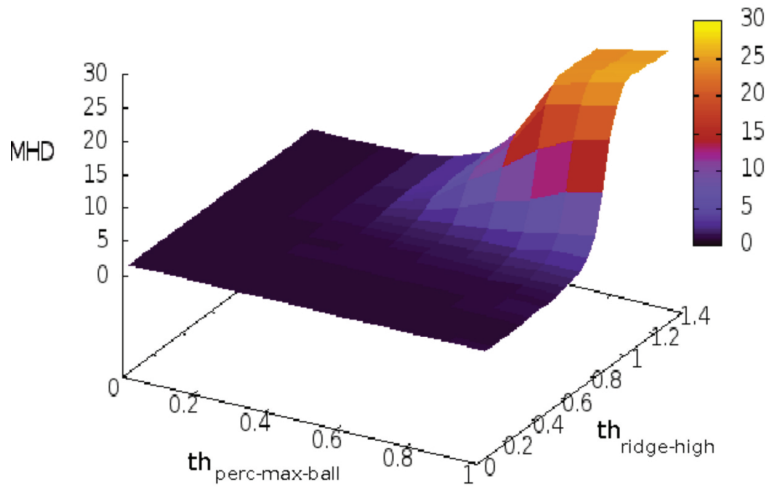
- Homotopie : la topologie du squelette doit correspondre à la celle de la forme. Les branches doivent être d'un pixel d'épaisseur. Les discontinuités le long des branches doivent être évitées.
- Faible complexité : l'appariement de formes implique souvent l'analyse de nombreuses formes dans une période de temps minimale. Une complexité linéaire est donc fortement souhaitable.



(a)



(b)



(c)

FIGURE 3.20 – *Variation des mesures Perc (a), HD (b) et MHD (c) en fonction des seuils* $th_{perc-max-ball}$ *et* $th_{ridge-high}$ *de DECS.*

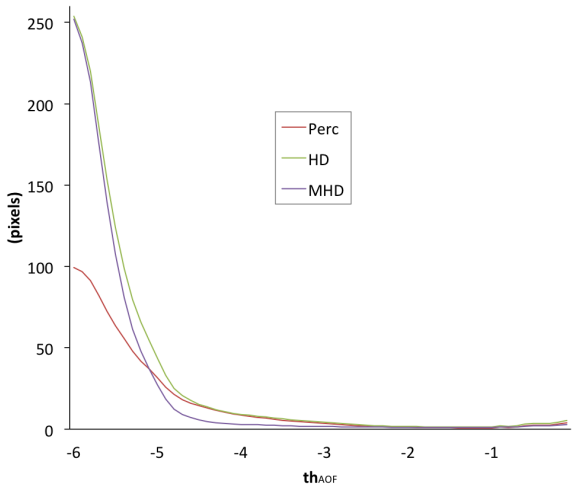


FIGURE 3.21 – *Variation des mesures Perc, HD et MHD en fonction du seuil th_{AOF} de la méthode de Siddiqi et al. (2002). Lorsque th_{AOF} diminue, les branches les moins importantes disparaissent.*

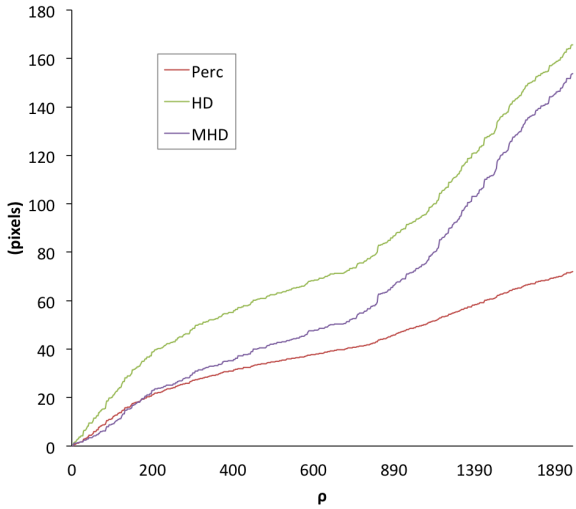


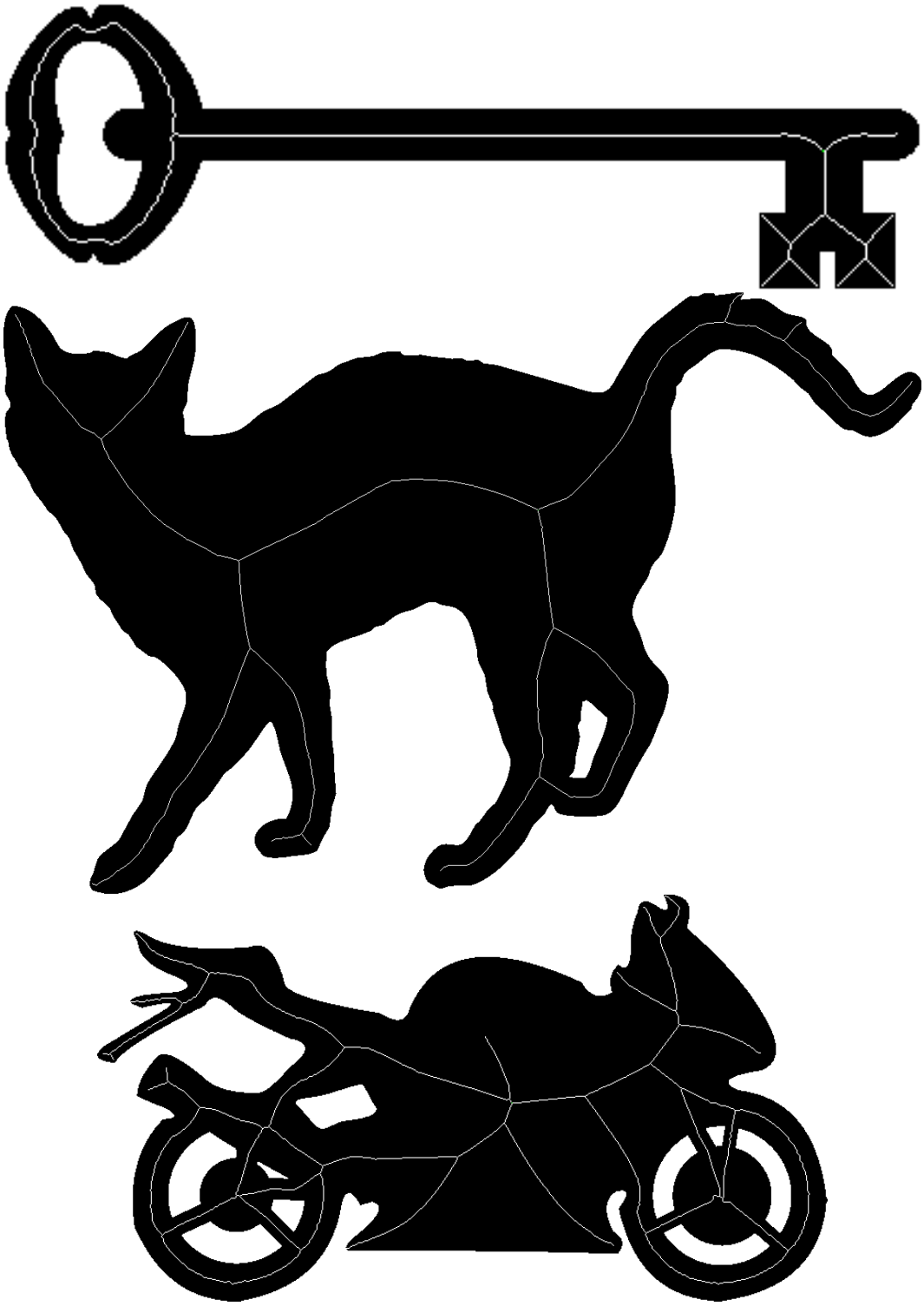
FIGURE 3.22 – *Variation des mesures Perc, HD et MHD en fonction du seuil ρ de la méthode de Choi et al. (2003). Lorsque ρ augmente, les branches les moins importantes disparaissent.*

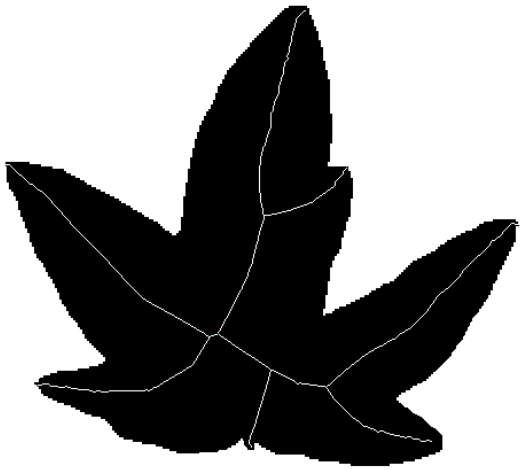
– Précision de la reconstruction : le squelette doit rendre compte, le plus fidèlement possible, de la forme et donc pouvoir reconstruire au mieux cette dernière.
– Résistance au bruit : cela permet d'éliminer les branches inutiles afin d'éviter de surcharger le squelette pour gagner du temps lors de l'appariement.

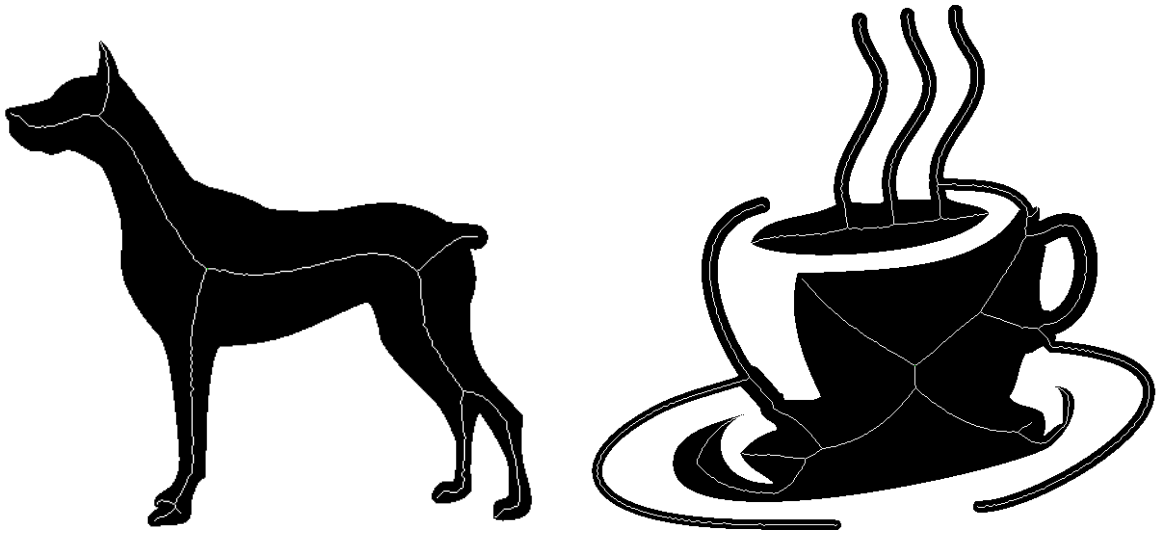
En ce qui concerne la robustesse au bruit, nous voyons une nette supériorité des algorithmes basés sur une carte de distance. Cette supériorité n'est pas confirmée pour d'autres critères étant donné que les principaux inconvénients de la méthode de Siddiqi et al. (2002) sont sa complexité, et surtout de son manque de résistance au bruit. Le principal problème de la méthode de Choi et al. (2003) est la perte de connexité du squelette lorsque le seuil augmente, c'est-à-dire lorsque les branches inutiles ne sont pas prises en compte (ce qui est un aspect essentiel pour nous).

3.4 EXEMPLES DE SQUELETTES

L'ENSEMBLE de ces squelettes ont été obtenus en utilisant les seuils suivants : $th_{perc-max-ball}$=0.4 et $th_{ridge-high}$= 0.5.







Conclusion du chapitre

Dans ce chapitre, nous avons présenté un algorithme linéaire pour extraire un Squelette Euclidien Discret Connecté d'une forme (Leborgne et al. 2014a;b; 2015). Pour ce faire, nous avons proposé un algorithme de propagation sur les crêtes de la carte de distance euclidienne et les centres de boules maximales. La propagation commence à partir du centre de la boule maximale ayant le plus grand rayon, ce qui assure la connexité. Nous obtenons alors un squelette fin grâce à l'algorithme d'amincissement *MB*2 (Manzanera et al. 1999, Bernard et Manzanera 1999) et au post-traitement qui lui est ajouté. La dernière étape consiste à réduire le squelette en procédant à l'élagage des branches, sur la base d'un critère utilisant simultanément les valeurs de crête et les centres des boules maximales.

Le squelette proposé a des propriétés souhaitables telles que la connexité, la finesse et la robustesse au bruit dans le cadre d'une utilisation pour l'appariement de formes. À notre connaissance, aucun algorithme de squelettisation, précédemment rapporté dans la littérature, n'est robuste au bruit tout en préservant la connexité et la topologie de la forme. Notons que le squelette proposé est également calculé en temps linéaire et permet de reconstruire la forme avec précision.

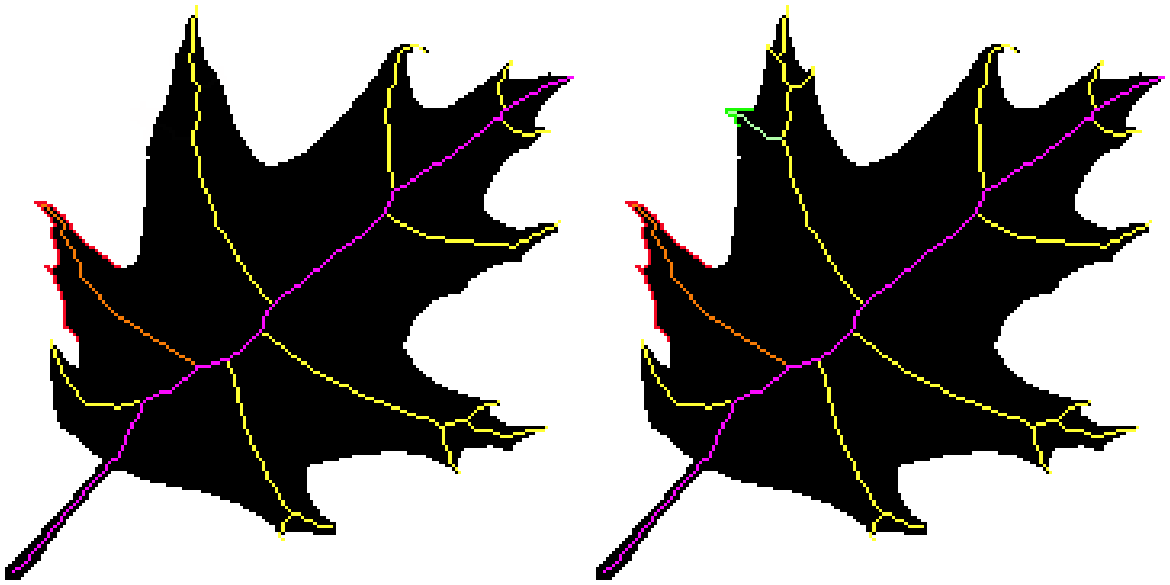
Il faut noter que l'algorithme de propagation utilisé pour générer un squelette connecté est utilisable sur les formes connexes. Pour les formes à composantes multiples, il suffit de détecter chacune des composantes connexes présentes dans l'image et appliquer notre méthode sur chacune d'elles.

L'étape suivante consiste à hierarchiser les branches du squelette en fonction de leur importance dans la représentation de la forme.

HIÉRARCHISATION DU SQUELETTE 4

SOMMAIRE

4.1 LISSAGE À AIRE CONSTANTE . . . 103
4.1.1 Équation d'évolution (Gage (1986),Dolcetta et al. (2002)) . . . 103
4.1.2 Discrétisation . . . 104
4.1.3 Ré-échantillonnage . . . 105
4.1.4 Obtention d'une forme discrète à partir du contour lissé . . . 106
4.1.5 Remplissage de la Forme Discrète . . . 106
4.2 MÉTHODES POUR LA CONSTRUCTION D'UN SQUELETTE HIÉRARCHIQUE . . . 106
4.3 MODÉLISATION DU SQUELETTE HIÉRARCHIQUE PAR UN HYPER-GRAPHE HIÉRARCHIQUE . . . 118
4.3.1 Définition . . . 118
4.4 RÉSULTATS . . . 119
4.4.1 Stabilité sous transformations affines . . . 119
4.4.2 Élagage du squelette . . . 121
4.4.3 Complexité . . . 123
CONCLUSION . . . 123



(a) (b)

FIGURE 4.1 – *Illustration des branches du squelette, qui sont plus ou moins importantes. (a) Squelette d'une feuille, (b) Squelette de cette même feuille avec quelques détails en plus.*

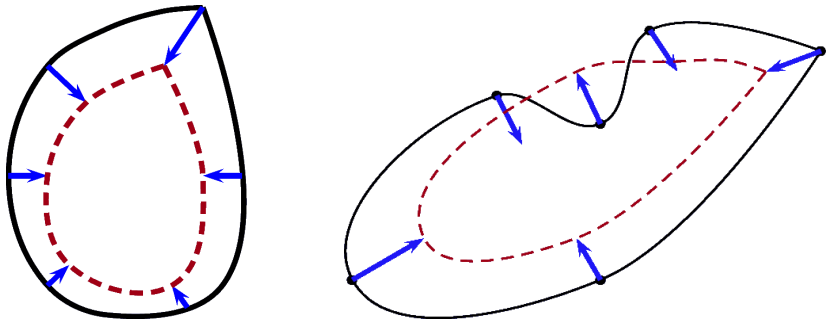
UN squelette est une représentation concise de la forme. Pour faire de l'appariement, un squelette idéal est homotope à la forme, fin (un pixel d'épaisseur), robuste au bruit et permettant la reconstruction de la forme. Dans le chapitre précédent, nous avons proposé le Squelette Connecté Euclidien Discret (DECS), qui remplit ces exigences (Leborgne et al. (2015)).

Dans ce chapitre nous nous autorisons à manipuler une concaténation de branches de squelette, pourvu qu'il n'y ait toujours que deux extrémités et qu'une même branche ne soit présente que dans une seule concaténation. Nous verrons dans ce chapitre comment les construire, mais avant il est nécessaire de comprendre notre intuition. Lors de l'appariement de deux formes utilisant leur squelette, il semble naturel d'apparier les branches (ou concaténations de branches) ayant le même ordre d'importance. Considérons la concaténation de branches la plus importante d'une première forme, c'est-à-dire celle qui a été générée par l'allure globale de la forme (un exemple étant la concaténation de branches représentée en fuchsia sur la Figure 4.1(a)). De toute évidence, il n'y a aucune raison de l'apparier à une branche (ou concaténation de branches) de la seconde forme ayant une faible importance, c'est-à-dire une branche (ou concaténation de branches) provenant d'un détail de la forme (par exemple la branche verte sur la Figure 4.1(b)). En effet, un détail de la forme est conceptuellement très différent de l'allure générale de la forme.

La quantification de l'importance de chaque branche (ou concaténation de branches) peut être appliquée aux squelettes car chacune provient d'une irrégularité plus ou moins importante du bord de la forme. Considérons l'exemple de la feuille d'arbre sur la Figure 4.1. Visuellement, les deux images sont très similaires puisque seulement trois petits détails ont été ajoutés à la Figure 4.1(b) par rapport à la Figure 4.1(a). Néanmoins, les deux squelettes obtenus sont sensiblement différents et pourtant, nous voudrions apparier ces deux formes. Par conséquent,

les branches (ou concaténations de branches) correspondant à des détails devraient avoir une importance très faible, pour obtenir deux squelettes similaires ou presque. Plus précisément, la composante verte du bord est à l'origine de la branche de squelette verte. De la même manière, la composante rouge du bord est responsable de la branche de squelette orange. Ainsi, la branche de squelette orange est plus importante que la verte.

Pour déterminer l'importance des branches (ou concaténations de branches), nous proposons de lisser successivement la forme (Section 4.1). Ainsi, les détails du bord sont supprimés et les branches de squelette éliminées. Nous proposons deux méthodes de mise à jour du squelette : une méthode naïve et une méthode de squelette déformable (Section 4.2). Pour quantifier l'importance des branches, nous avons opté pour l'échelle à laquelle les branches disparaissent lorsque nous effectuons plusieurs lissages successifs. Ce faisant, le nombre de branches décroît lorsque le bord est lissé. En conséquence, plus une branche de squelette persiste, plus elle est importante.



(a) (b)

FIGURE 4.2 – *Illustration de l'évolution de deux courbes sous l'effet du flot de courbure moyenne.*

4.1 LISSAGE À AIRE CONSTANTE

SOIT $\mathcal{F}^{(0)} \subset \mathbb{Z}^2$ la forme initiale. Nous souhaitons générer une séquence $(\mathcal{F}^{(0)}, \mathcal{F}^{(1)}, ..., \mathcal{F}^{(\lambda)}, ...)$ de formes successivement lissées. La conservation de l'aire permet le maintien des proportions tout au long du processus de lissage. La forme doit être lissée sans rétrécir car nous voulons extraire des squelettes sur des formes lissées de même taille. Dorénavant, nous traitons uniquement des formes connectées sans trou. Les feuilles d'arbre, qui sont la principale application visée, sont dans ce cas là.

4.1.1 Équation d'évolution (Gage (1986),Dolcetta et al. (2002))

À partir d'une liste de pointels, extraite par l'algorithme du suivi de contours présenté dans le Chapitre 1, représentant le contour de la forme initiale à l'aide d'une courbe fermée, notre objectif est d'obtenir des courbes fermées lissées successivement tout en conservant l'aire initiale de la forme.

Soient $C(s,t) = (x(s,t), y(s,t))$, la courbe variant en fonction du temps t et d'un paramètre spatial s, $\mathcal{N}(s,t)$ la normale unitaire intérieure de C, $\mathcal{L}$ la longueur euclidienne totale et $\kappa(s,t)$ la courbure euclidienne définie par :

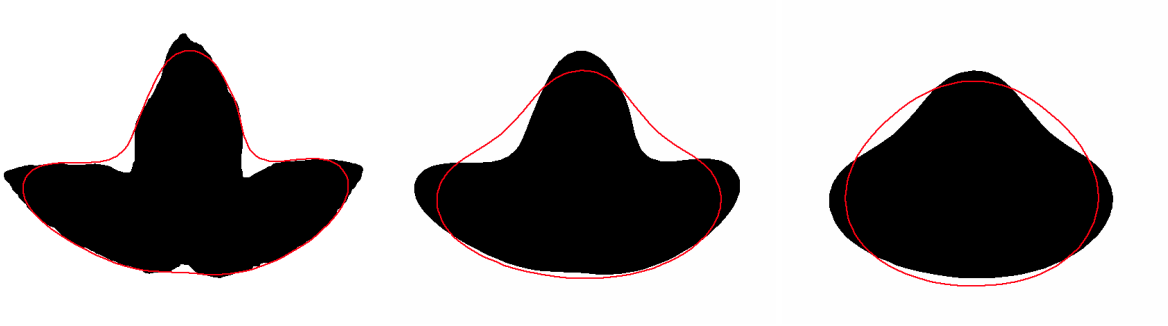
$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \tag{4.1}$$

où $'$ représente la dérivée partielle en fonction de s. La courbure euclidienne au point $\boldsymbol{p}$ de la courbe n'est autre que l'inverse du rayon angulaire de courbure. Ce dernier est la valeur du rayon du cercle tangent à la courbe en $\boldsymbol{p}$ dont le signe varie en fonction de la concavité ou convexité de la courbe. Ainsi, plus le rayon du cercle est grand et moins la courbure est importante.

Sous l'effet du flot de courbure moyenne

$$\frac{\partial C}{\partial t} = \kappa \mathcal{N}$$

la courbe se rétracte et tend vers un point, comme illustré sur la Figure 4.2.



(a) (b) (c)

FIGURE 4.3 – *Illustration de l'évolution d'une courbe (contour de la forme de la Figure (a)) sous l'effet du flot de l'Équation 4.2. À chaque évolution, une forme est créée à partir de la courbe précédemment lissée.*

Il est à noter que si on lisse une courbe indéfiniment, la forme finale de la courbe sera un cercle. Autrement dit, la courbe est asymptotiquement circulaire.

Pour réaliser ce lissage en conservant l'aire à l'intérieur de la courbe, nous avons utilisé l'équation suivante définie par Gage (1986) :

$$\frac{\partial C}{\partial t} = \left(\kappa - \frac{2\pi}{\mathcal{L}}\right)\mathcal{N} \tag{4.2}$$

Sous l'effet de ce flot, la courbe converge vers un cercle dont l'aire est égale à celle de la forme initiale, comme l'illustre la Figure 4.3.

4.1.2 Discrétisation

Les formes devant être définies dans l'espace discret, il est nécessaire de discrétiser l'Équation 4.2. Pour ce faire, nous avons estimé les dérivées par différences finies centrées :

$$x'(s) \approx \frac{x(s+\delta s) - x(s-\delta s)}{2\delta s} \text{ avec } \delta s = 1$$

$$x''(s) \approx \frac{x(s+\delta s) - 2x(s) + x(s-\delta s)}{\delta s^2} \text{ avec } \delta s = 1$$

et de même pour y. Nous avons choisi $\delta s = 1$ car nous supposons que la distance entre deux points consécutifs de la courbe est approximativement 1. La courbe étant discrétisée en une séquence de points $\boldsymbol{q}_i$, les premier et second ordres sont estimés par :

$$x'_{q_i} \approx \frac{-x_{q_{i-1}} + x_{q_{i+1}}}{2} \; et \; x''_{q_i} \approx x_{q_{i+1}} + x_{q_{i-1}} - 2x_{q_i}$$

et de la même façon pour y.

La longueur, la courbure et le vecteur normal sont estimés à l'aide du schéma de discrétisation précédent. La longueur $\mathcal{L}$ est calculée comme étant la somme des distances euclidiennes entre les points successifs de la courbe. $\mathcal{N}_i$ est calculé grâce à :

$$\mathcal{N}_i = \frac{(-y'_{q_i}, x'_{q_i})}{\sqrt{x'^2_{q_i} + y'^2_{q_i}}} \tag{4.3}$$

En résumé, pour une itération, chaque point de la courbe $\boldsymbol{q}_i$ est remplacé par le point obtenu avec l'équation d'évolution discrète (4.4), résultat de la discrétisation de l'Équation 4.2 avec un pas temporel Δt permettant de discrétiser le temps.

$$\boldsymbol{q}_i^{t+1} = \boldsymbol{q}_i^t + \Delta t \left(\kappa_i - \frac{2\pi}{\mathcal{L}} \right) \mathcal{N}_i \tag{4.4}$$

Dans cette équation, Δt est compris entre 0 et 1. Lorsqu'il est trop petit, la convergence est lente. Quand il est trop grand, une instabilité s'installe car les points oscillent. Nous avons donc choisi $\Delta t = 0,5$ dans la suite de nos travaux. Nous notons que λ, le nombre de lissages, est un sous-échantillonnage de t. Entre deux itérations λ et $\lambda + 1$, l'exécution représentée dans l'Equation (4.4) est effectuée plusieurs fois pour tous les points de la courbe. En effet, entre λ et $\lambda + 1$, nous voulons que les points de la courbe aient évolué suffisamment avant d'effectuer l'étape qui sera présentée dans la Section 4.2 et lors de laquelle la construction du squelette hiérarchique se doit d'être efficace. Pour ce faire, nous avons fait en sorte que le lissage soit perceptuellement "à vitesse constante". Lors de nos expérimentations, nous avons remarqué que, dans les premières étapes de lissage, la forme évoluait rapidement et lorsque le nombre de lissages augmentait, la forme bougeait plus lentement. Notre intuition a donc été d'accélérer le lissage pour qu'il paraisse linéaire. Le nombre de lissages réellement effectué est donc :

$$nb\_lissages\_reel(\lambda) = \frac{\lambda(\lambda + 1)}{2}$$

Notons qu'il est inutile de conserver toutes les itérations de lissage pour créer la séquence $(\mathcal{F}^{(0)}, ..., \mathcal{F}^{(\lambda)}, ...)$.

4.1.3 Ré-échantillonnage

Pour préserver la régularité du lissage avec une densité similaire à celle du contour initial, il est nécessaire de ré-échantillonner la courbe à chaque itération de lissage. Cela permet d'avoir une courbe homogène et une bonne approximation de la courbure, des normales, *etc.* Dans le cas contraire, l'estimation de la courbure deviendrait instable lorsque deux points seraient trop proches notamment. Lorsque deux points consécutifs de la courbe sont trop proches, ils sont fusionnés en un point au milieu. De même, nous ajoutons un point au milieu de deux points consécutifs quand ceux-ci sont trop éloignés.

L'objectif est de maintenir la condition suivante :

$$a \leq \left\| \boldsymbol{q}_i - \boldsymbol{q}_{i+1} \right\| \leq 2a$$

où a appartient à $\mathbb{R}$. Dans nos travaux, nous avons choisi $a = 1$.

La Figure 4.4 montre un exemple d'une partie du contour d'une forme avant et après le ré-échantillonnage.

Nous notons qu'une passe de ré-échantillonnage est effectuée après chaque passe de lissage.

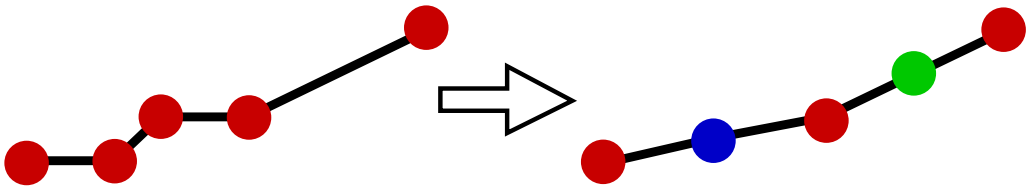


FIGURE 4.4 – *Exemple de ré-échantillonnage du contour d'une forme. Le point bleu provient d'une fusion, alors que le point vert résulte d'une séparation.*

4.1.4 Obtention d'une forme discrète à partir du contour lissé

Comme DECS (Leborgne et al. (2015)) ne peut être appliqué que sur une forme représentée par un sous-ensemble de $\mathbb{Z}^2$, il est nécessaire de convertir le contour lissé en une séquence de points appartenant à $\mathbb{Z}^2$. Le bord de la forme est ici une ligne brisée constituée de segments continus. Les points, extrémités de ces segments, sont alors discrétisés en arrondissant leurs coordonnées à l'entier le plus proche. Puis, un segment discret est tracé entre chaque paire de points consécutifs $[\boldsymbol{q}_i, \boldsymbol{q}_{i+1}]$ grâce à l'algorithme de Bresenham, présenté dans la Section 1.5 du Chapitre 1.

4.1.5 Remplissage de la Forme Discrète

Comme précédemment, nous considérons qu'aucun pixel de la forme ne se situe sur le bord de l'image.

Par conséquent, pour obtenir la forme discrète à partir du contour discret, nous remplissons l'extérieur et considérons le complémentaire de la zone remplie comme la forme. Pour ce faire, nous marquons un point du bord de l'image $\boldsymbol{q}_b$ comme appartenant au fond puis nous propageons ce marquage à tous les pixels $\boldsymbol{q}$ voisins 4-connexes entre $\boldsymbol{q}_b$ et $\boldsymbol{q}$ sans point de contour. Tous les pixels restants sont marqués comme appartenant à la forme. Ces étapes sont représentées sur la Figure 4.5.

Il est à noter qu'à partir de ce chapitre, nous traitons uniquement les formes sans trou. C'est pourquoi un remplissage par l'extérieur est suffisant.

4.2 MÉTHODES POUR LA CONSTRUCTION D'UN SQUELETTE HIÉRARCHIQUE

L'IDÉE générale est d'obtenir un squelette pour chaque forme lissée et de les condenser tous en un squelette hiérarchique. Dorénavant, nous appellerons n_{smooth_max} le nombre de lissages nécessaires pour réduire le squelette à un point (et de manière équivalente, pour réduire la forme à un cercle) grâce à une succession de lissages.

Dans la suite de cet article, nous considérons qu'un squelette est un ensemble de branches ayant des relations d'adjacence, comme défini précisément dans le Chapitre 3.

Pour plus de commodité, le vocabulaire concernant les branches et les points de squelette est illustré sur la Figure 4.6. Les points $\boldsymbol{p}_1$ à $\boldsymbol{p}_6$ sont des extrémités de branches. Plus précisé-

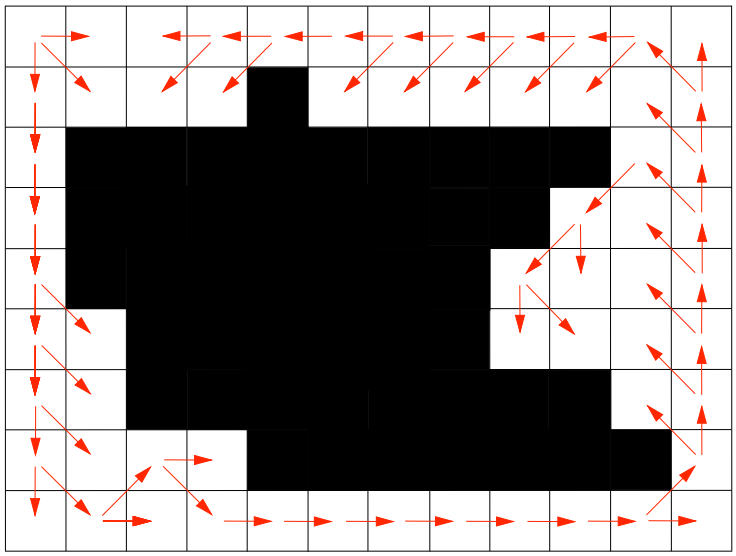


FIGURE 4.5 – *Exemple du coloriage d'une forme à partir de son contour.*

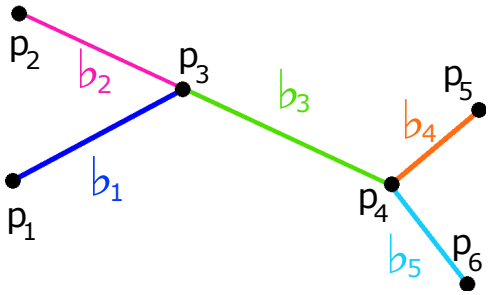


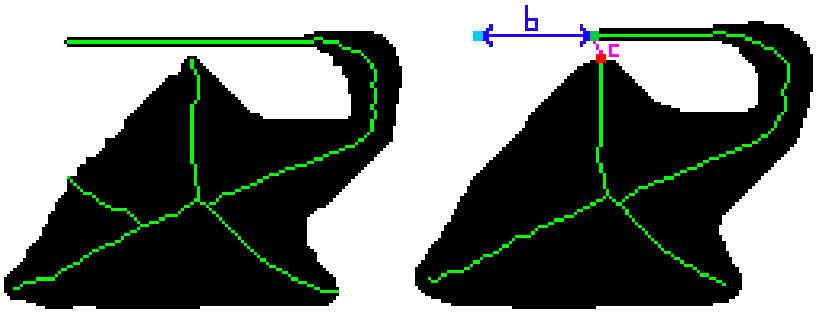
FIGURE 4.6 – *Illustration du vocabulaire utilisé pour les branches et les points de squelette.*

ment, $\boldsymbol{p}_3$ et $\boldsymbol{p}_4$ sont des points de jonction alors que $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_5$ et $\boldsymbol{p}_6$ sont des points terminaux. $\mathfrak{b}_1$ à $\mathfrak{b}_5$ sont des branches composées de deux points extrémité, avec, entre ces deux points, des points de branche. Plus précisément, $\mathfrak{b}_1$, $\mathfrak{b}_2$, $\mathfrak{b}_4$ et $\mathfrak{b}_5$ sont des branches terminales alors que $\mathfrak{b}_3$ est une branche interne. Nous appelons $S_{\mathcal{F}^{(\lambda)}}$, le squelette calculé à partir de la forme lissée $\mathcal{F}^{(\lambda)}$.

Définition 4.1 (Méta-branche) *Une méta-branche $\mathfrak{b}^m$ est une séquence (ou concaténation) de branches $(\mathfrak{b}_0, ..., \mathfrak{b}_{u'-1})$ appartenant à $S_{\mathcal{F}^{(0)}}$ tel que pour tout $i = 0, ...(u'-2)$, $\mathfrak{b}_i$ et $\mathfrak{b}_{i+1}$ ont un pixel extrémité en commun. Une méta-branche a deux extrémités.*

Définition 4.2 (Squelette hiérarchique) *Le squelette hiérarchique, noté S_h, est un ensemble de méta-branches pondérées tel que toutes les branches de $S_{\mathcal{F}^{(0)}}$ appartiennent à une méta-branche. Le poids d'une méta-branche $\mathfrak{b}^m$ est indiqué par $w_{\mathfrak{b}^m}$.*

Pour construire un squelette hiérarchique S_h, nous proposons d'abord une méthode dite "naïve", qui consiste à recalculer le squelette pour chaque forme lissée. Cependant, les squelettes obtenus ne sont pas suffisamment stables comme nous le verrons dans les exemples. C'est la raison pour laquelle nous avons implémenté une autre méthode basée sur une défor-



(a) $S_{\mathcal{F}(\lambda-1)}$ (b) $S_{\mathcal{F}(\lambda)}$

FIGURE 4.7 – *Exemple d'appariement non trivial d'un point terminal de squelette entre deux étapes consécutives de lissage.*

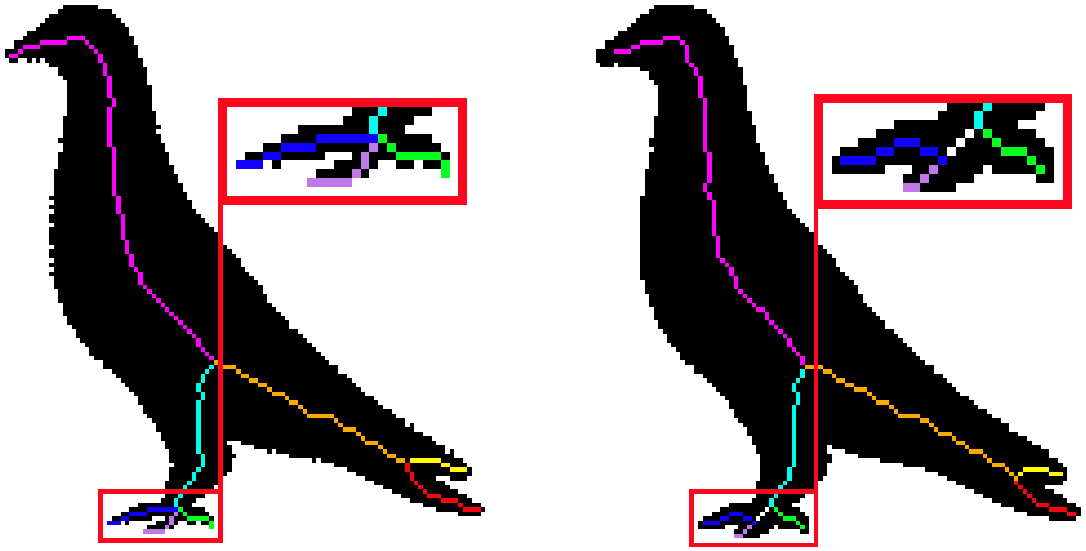
mation du squelette initial dans le but de l'adapter successivement aux formes lissées. Il s'agit de la méthode appelée "squelette déformable".

Méthode naïve

Le principe de la méthode naïve est de recalculer un squelette avec DECS sur chaque forme lissée jusqu'à ce que le squelette se résume à seulement un point. Ainsi, une séquence de squelettes $h(S) = (S_{\mathcal{F}(\lambda)})_{\lambda \in [0; n_{smooth\_max}]}$ est générée. Pour construire le squelette hiérarchique à partir de $h(S)$, chaque branche terminale de chaque squelette $S_{\mathcal{F}(\lambda)}$ doit être appariée avec une branche du squelette d'origine $S_{\mathcal{F}(0)}$. Le but est de déterminer, pour chaque branche $\flat$ de $S_{\mathcal{F}(0)}$, l'itération à laquelle $\flat$ disparaît, c'est-à-dire l'itération à laquelle aucune branche ne s'apparie avec $\flat$. Les branches de $S_{\mathcal{F}(\lambda)}$ ne sont pas identiques à celles de $S_{\mathcal{F}(0)}$. Souvent, elles se raccourcissent jusqu'à disparaître. Entre deux lissages consécutifs, un point terminal sur $S_{\mathcal{F}(\lambda-1)}$ peut être plus proche, en terme de distance euclidienne, d'un point terminal situé sur une branche voisine sur $S_{\mathcal{F}(\lambda)}$ que du point terminal de sa propre branche. C'est le cas des branches issues de fines protubérances de la forme, qui sont lissées rapidement. Par exemple sur la Figure 4.7, la distance c est plus petite que la distance b entre deux points de squelette correspondants. De plus, les branches internes peuvent être réorganisées, comme le montre la Figure 4.8 et, dans certains cas, assez rares, des branches peuvent apparaître.

L'appariement n'est donc pas trivial. Nous avons développé l'Algorithme 7 pour le résoudre. La première étape est l'initialisation du squelette hiérarchique S_h (l 2-10 de l'Algorithme 7). Les méta-branches de S_h sont les branches de $S_{\mathcal{F}(0)}$. A chaque méta-branche sont associés :

- w : l'itération à laquelle la méta-branche devient inutile. Au début du traitement, toutes les méta-branches sont présentes, donc $w = -1$ pour chaque méta-branche (par convention).
- *end_points* : l'ensemble des points terminaux (au maximum 2) de la méta-branche (s'ils existent). Nous pouvons noter que, dans un squelette hiérarchique, les méta-branches terminales ont généralement un point terminal tandis que la méta-branche principale peut avoir deux points terminaux.



(a) $S_{\mathcal{F}(\lambda-1)}$ (b) $S_{\mathcal{F}(\lambda)}$

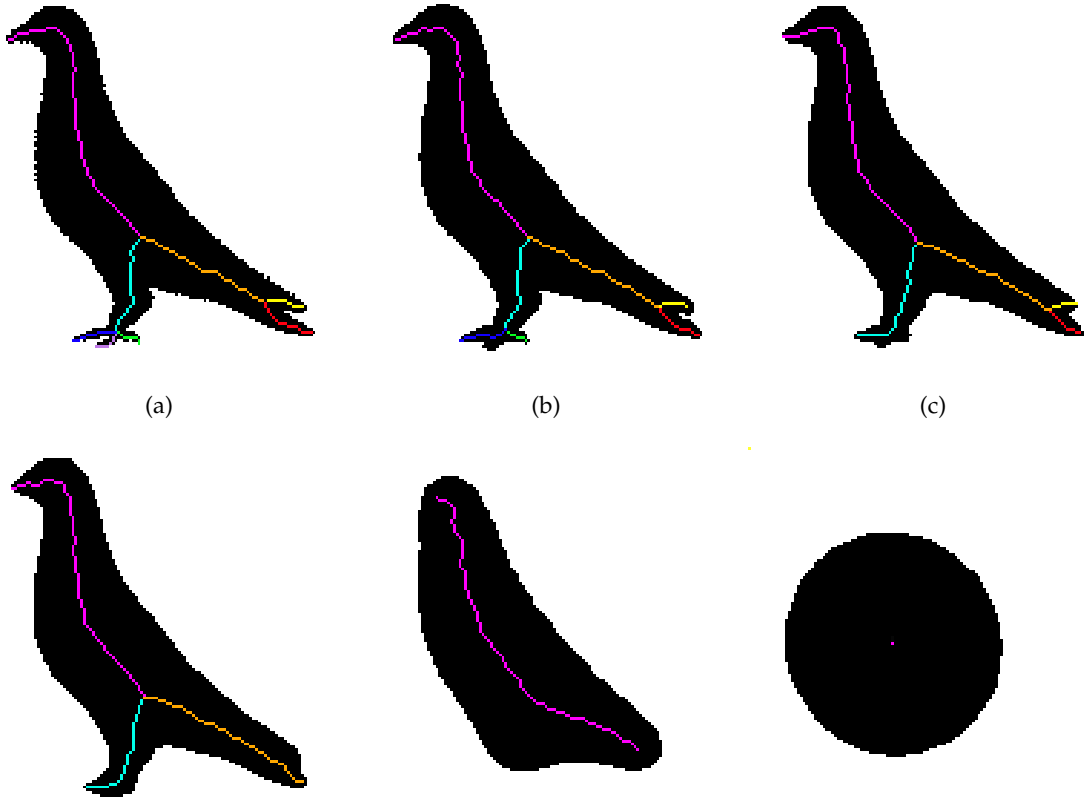
FIGURE 4.8 – *Exemple de réorganisation interne du squelette entre deux étapes consécutives de lissage.*

La deuxième étape est l'appariement des branches avec les méta-branches. En effet, pour chaque squelette $S_{\mathcal{F}(\lambda)}$ de $h(S)$, nous apparions les branches terminales de $S_{\mathcal{F}(\lambda)}$ avec les méta-branches terminales du squelette hiérarchique de façon bidirectionnelle, afin de minimiser les erreurs dues aux remarques précédentes. Ceci est décrit aux lignes 13 à 21 de l'Algorithme 7. Les méta-branches terminales de S_h, qui ne sont pas appariées à la fin de l'étape précédente, sont étiquetées comme étant inutiles à partir de l'itération λ (l 22-27 de l'Algorithme 7). En d'autres termes, le poids w des méta-branches concernées est remplacé par λ.

Finalement, le squelette hiérarchique est réorganisé pour conserver une structure de squelette lorsque nous ignorons les méta-branches considérées comme inutiles à cette étape (celles qui ont un poids supérieur à -1). La Figure 4.9 illustre ce qui se passe tout au long des étapes de lissage pour obtenir un squelette hiérarchique (Fig. 4.10).

Algorithme 7: Pseudo-code de la méthode dite naïve.

Entree : $h(S)$, **Sortie :** S_h
*/*Initialiser*/*
Ajouter toutes les branches de $S_{\mathcal{F}^{(0)}}$ *à* S_h
pour tous les *branches* $\flat$ *de* S_h **faire**
    $w_\flat := -1$
    si $\flat$ *est une branche terminale* **alors**
        Ajouter les points terminaux de $\flat$ *à* $\flat$*.end_points*
    fin
fin
*/*Apparier les branches terminales de* $h(S)$ *avec celles de* S_h**/*
pour λ *allant de* 1 *à* n_{smooth_max} **faire**
    pour tous les *points terminaux* $\boldsymbol{p}_j$ *de* $S_{\mathcal{F}^{(\lambda)}}$ **faire**
        $\boldsymbol{p}_{S_h} := \underset{\boldsymbol{p}_k \in \cup_{\flat \in S_h} \flat.end\_points}{\operatorname{argmin}} d(\boldsymbol{p}_j, \boldsymbol{p}_k)$
        $\boldsymbol{p}_{S_{\mathcal{F}^{(\lambda)}}} := \underset{\boldsymbol{p}'_j \in points\ terminaux\ de\ S_{\mathcal{F}^{(\lambda)}}}{\operatorname{argmin}} d(\boldsymbol{p}'_j, \boldsymbol{p}_{S_h})$
        si $\boldsymbol{p}_j = \boldsymbol{p}_{S_{\mathcal{F}^{(\lambda)}}}$ **alors**
            Marquer comme nécessaires, les branches
            terminales concordant avec $\boldsymbol{p}_{S_h}$ *, remplacer le*
            point terminal concordant avec $\boldsymbol{p}_{S_h}$ *par* $\boldsymbol{p}_{S_{\mathcal{F}^{(\lambda)}}}$
        fin
    fin
    */*Supprimer les branches terminales vaines à l'itération* λ**/*
    pour tous les *branches terminales* $\flat$ *de* S_h **faire**
        si $\flat$ *n'est pas marquée et* $w_\flat = -1$ **alors**
            $w_\flat = \lambda$
        fin
    fin
    */*Réorganiser le squelette hiérarchique*/*
    pour tous les *branches terminales* $\flat$ *de* S_h **faire**
        pour tous les *points extrémités de* $\flat$ **faire**
            */*Illustration sur la Figure 4.11*/*
            si $|\mathcal{B}_A(\flat, \boldsymbol{p})| = 1$ **alors**
                Fusionner $\flat$ *avec sa branche adjacente au point* $\boldsymbol{p}$
            fin
            */*Illustration sur la Figure 4.12*/*
            si $|\mathcal{B}_A(\flat, \boldsymbol{p})| = 0$ **alors**
                Ajouter $\boldsymbol{p}$ *à* $\flat$*.end_points*
            fin
        fin
    fin
fin
retourner S_h



(a) (b) (c)

(d) (e) (f)

FIGURE 4.9 – *Hiérarchisation du squelette : (a) squelette de la forme d'origine $S_{\mathcal{F}^{(0)}}$, (b) $S_{\mathcal{F}^{(6)}}$, suppression de la branche violette, (c) $S_{\mathcal{F}^{(15)}}$, suppression de la branche verte et fusion des branches bleue et turquoise, (d) $S_{\mathcal{F}^{(21)}}$, suppression de la branche jaune et fusion des branches rouge et orange, (e) $S_{\mathcal{F}^{(210)}}$, suppression de la branche turquoise et fusion des branches orange et rose, (f) $S_{\mathcal{F}^{(4015)}}$, réduction de la branche principale à un point.*

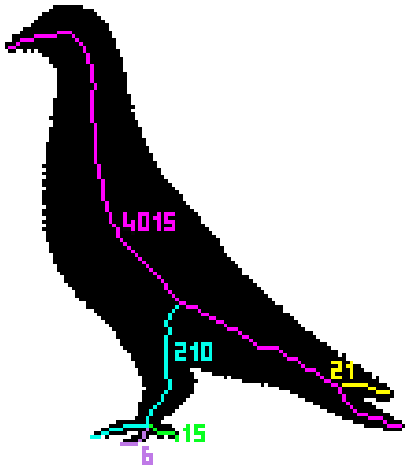
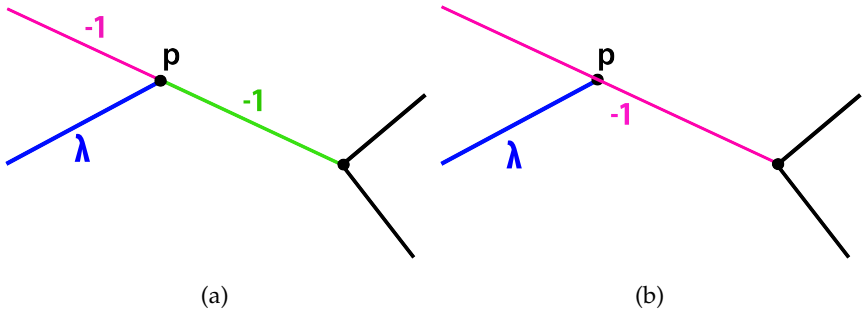


FIGURE 4.10 – *Squelette hiérarchique S_h obtenu à partir de l'évolution représentée sur la Figure 4.9. Les étiquettes des méta-branches sont les nombres de lissages nécessaires pour la supprimer.*



(a) (b)

FIGURE 4.11 – *Squelette hiérarchique avant (a) et après (b) la mise à jour.* $\boldsymbol{p}$ *est l'endroit où les branches rose et verte sont concaténées. Les métabranches sont étiquetées avec leur poids.*

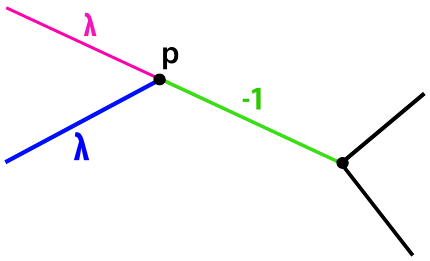


FIGURE 4.12 – *Cas où une méta-branche interne devient une méta-branche terminale durant la mise à jour du squelette hiérarchique. Par conséquent,* $\boldsymbol{p}$ *devient un point terminal. Les méta-branches sont étiquetées avec leur poids.*

Squelette déformable

Le but est identique à celui de la méthode naïve, à savoir, obtenir un squelette pour chaque forme lissée jusqu'à ce que ce dernier se résume à un point. Nous obtenons donc également une séquence de squelettes $h(S) = (S_{\mathcal{F}^{(\lambda)}})_{\lambda \in [0; n_{smooth\_max}]}$. À la différence de la méthode naïve, nous ne recalculons pas le squelette sur chaque forme mais nous déformons celui de l'itération précédente pour l'adapter à chaque forme lissée. Notre squelette déformable est basé sur le modèle de contour actif introduit par Kass et al. (1988). Le principe est de considérer une série de points déplaçables formant une courbe en deux dimensions (dans nos travaux, chaque branche du squelette constitue une courbe). Chacune d'elles se déplace alors pour aller se positionner sur la crête de la carte de distance la plus proche en essayant de conserver ses caractéristiques. Pour ce faire, deux énergies sont utilisées. L'objectif est alors de minimiser l'énergie totale présente le long de la courbe. La première énergie est appelée énergie interne. Elle ne dépend pas de la forme mais uniquement des points de la courbe. Elle doit empêcher un point de se déplacer trop loin du reste de la courbe. Ainsi, elle permet de conserver une courbe lisse composée de points équidistants tout en autorisant sa déformation qui est alors induite par l'énergie externe.

Nous considérons $\boldsymbol{b}(u,t)$, la paramétrisation continue d'une branche $\flat$ sous forme de courbe ouverte et régulière. Dans cette section, l'exposant (λ) est ignoré par commodité. Sauf mention contraire, toutes les quantités sont considérées à l'échelle courante λ. Le squelette déformable est initialisé à l'aide du squelette final de l'échelle précédente $\lambda - 1$. Les points du squelette étant déplacés plusieurs fois, entre l'itération λ et $\lambda + 1$, avant de se positionner cor-

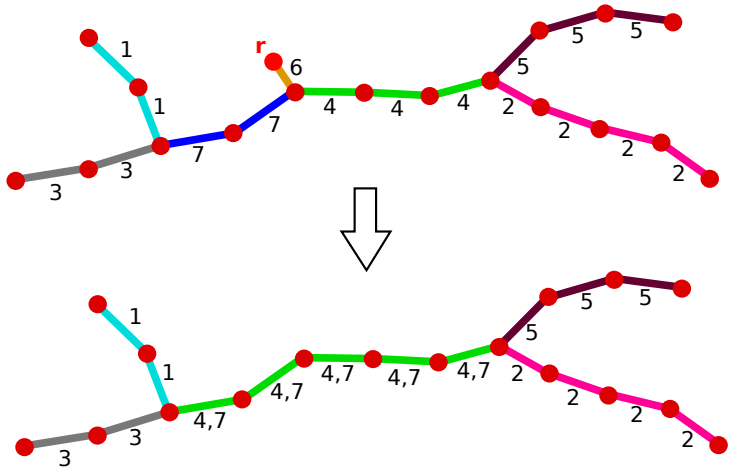


FIGURE 4.13 – *Propagation des étiquettes après suppression d'une branche : la suppression du sommet* **r** *provoque la disparition de la branche d'étiquette* 6*. Les deux branches adjacentes (respectivement étiquetées* 4 *et* 7*) sont fusionnées et les étiquettes sont combinées dans la branche fusionnée.*

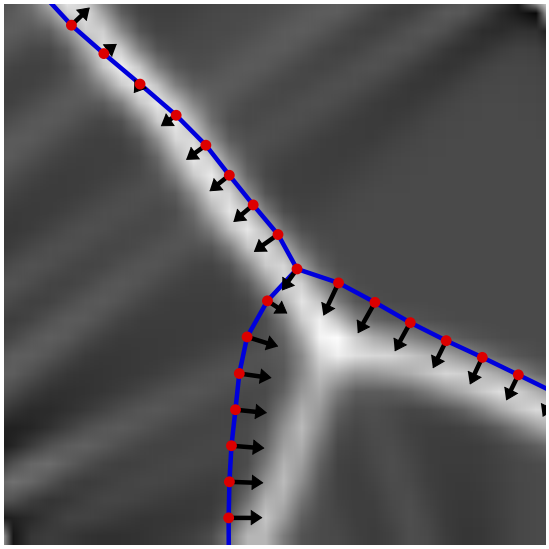


FIGURE 4.14 – *Forces attirant les sommets vers les crêtes de la carte de distance euclidienne*

rectement sur les crêtes, nous utilisons t, le temps d'évolution. Chaque branche évolue selon le flot géométrique suivant :

$$\begin{aligned} \frac{\partial \boldsymbol{b}(u,t)}{\partial t} &= (\nabla EDT(\boldsymbol{b}(u,t)) \cdot \mathcal{N}(u,t))\,\mathcal{N}(u,t) + \gamma \frac{\partial^2 \boldsymbol{b}(u,t)}{\partial u^2} \\ \boldsymbol{b}(\cdot,0) &= \boldsymbol{b}^{(\lambda-1)}(\cdot) \end{aligned} \tag{4.5}$$

où $\mathcal{N}$ est la normale à la courbe et γ est le poids contrôlant l'importance de la force de régularisation par rapport à la force d'attache aux données. En d'autres termes, cette force impose des contraintes sur la géométrie lors de la déformation. La force d'attache aux données attire les branches du squelette vers les crêtes les plus proches. Pour ce faire, le champ de vecteurs ∇EDT, dirigé vers les crêtes de la carte de distance euclidienne, est projeté localement sur la normale à la courbe. Ainsi, la branche est encouragée à s'aligner sur la crête la plus proche.

Sommets et étiquetage des arêtes Le squelette est échantillonné en un arbre. La position du $i^{\text{ième}}$ sommet est notée $\boldsymbol{\varrho}_i \in \mathbb{R}^2$, l'ensemble des indices de ses sommets adjacents η_i. Le $i^{\text{ième}}$ sommet est un point de branche si $|\eta_i| = 2$, un point terminal si $|\eta_i| = 1$ et un point de jonction si $|\eta_i| > 2$. Un point extrémité est un point de jonction ou un point terminal, en adéquation avec le vocabulaire illustré en Figure 4.6. Ici, une branche est une chaîne de sommets reliant deux sommets extrémités. Notons qu'une branche peut être composée uniquement de deux extrémités et de l'arête les liant (auquel cas elle ne possède pas de point de branche).

Les arêtes reliant les sommets sont étiquetées avec l'indice de la branche correspondante dans le squelette initial $S_{\mathcal{F}^{(0)}}$, illustré en Figure 4.13. Comme précisé plus loin, les étiquettes des branches initiales seront transférées aux branches fusionnées lors de la suppression des branches.

Déformation discrète (l. 5 à 12 de l'Algorithme 8) La force de régularisation est calculée comme suit :

$$\boldsymbol{r}_i = \begin{cases} \left(\frac{1}{|\eta_i|} \sum_{j \in \eta_i} \boldsymbol{\varrho}_j \right) - \boldsymbol{\varrho}_i & \text{si } |\eta_i| \geq 2 \\ \mathbf{0} & \text{sinon} \end{cases} \tag{4.6}$$

Ainsi, la force de régularisation a tendance à ramener un point de la courbe vers le centre de gravité de ses voisins, ce qui a pour effet de lisser le squelette. La force d'attache aux données nécessite le calcul de la normale $\mathcal{N}_i$ pour les points de branche, afin de les attirer vers la crête la plus proche. Elle est estimée à l'aide des deux sommets adjacents, par différences finies. Pour les extrémités, la normale considérée est celle de l'unique sommet adjacent. Pour les points de jonction, le gradient de la carte de distance euclidienne est pris tel quel, sans projection. Nous écrivons ainsi la force d'attache aux données :

$$\boldsymbol{f}_i = \begin{cases} \left(\mathcal{N}_j \cdot \nabla EDT(\boldsymbol{\varrho}_i)\right) \mathcal{N}_j & \text{si } |\eta_i| = 1, \eta_i = \{j\} \\ \left(\mathcal{N}_i \cdot \nabla EDT(\boldsymbol{\varrho}_i)\right) \mathcal{N}_i & \text{si } |\eta_i| = 2 \\ \nabla EDT(\boldsymbol{\varrho}_i) & \text{sinon} \end{cases} \tag{4.7}$$

L'équation d'évolution en chaque sommet, après discrétisation d'Euler avec un pas temporel Δt, est :

$$\boldsymbol{\varrho}_i^{(t+1)} = \boldsymbol{\varrho}_i^{(t)} + \Delta t \left(\chi \boldsymbol{r}_i^{(t)} + \boldsymbol{f}_i^{(t)} \right) \tag{4.8}$$

en prenant $\Delta t = 0.5$ et $\chi = 2$. Si χ est trop grand, l'énergie externe devient négligeable, ce qui empêche la courbe de se déformer et d'aller se positionner sur les crêtes, si elle ne l'est pas déjà. Expérimentalement, χ doit être compris entre 0,5 et 4.

Ré-échantillonnage et suppression des branches (l. 13 à 29 de l'Algorithme 8) Comme le contour décrit en Section 4.1, le squelette déformable est ré-échantillonné après chaque itération d'évolution, de manière à maintenir un espacement régulier entre les sommets. Pour chaque paire de sommets adjacents $(\boldsymbol{\varrho}_i, \boldsymbol{\varrho}_j)$, le ré-échantillonnage vise à ce que la condition

$c \leq \left\| \boldsymbol{\varrho}_i - \boldsymbol{\varrho}_j \right\| \leq 2c$ soit vérifiée (nous avons choisi $c = 2$). Si la distance entre deux sommets adjacents excède $2c$, un point de branche est créé entre les deux, quel que soit le degré des deux sommets. A l'inverse, si la distance entre deux sommets adjacents est inférieure à c, les degrés sont vérifiés. Si les deux sommets sont des points de branche, ils sont fusionnés, la nouvelle position étant le milieu des deux anciens sommets. Si l'un des sommets est terminal, il est supprimé. Ceci est en accord avec l'*a priori* que nous avons sur les déformations des crêtes de la carte de distance. En effet, au fur et à mesure des lissages, les crêtes de la carte de distance (et les branches de squelette correspondantes) sont obligatoirement raccourcies, quand elles ne sont pas supprimées. Le dernier cas concerne les points de jonction. Si l'un des sommets est un point de jonction et l'autre un point de branche, ce dernier est éloigné du premier de manière à ce qu'ils deviennent distants de c. Si les deux sommets sont des points de jonction, ils sont laissés en l'état.

Comme les crêtes sont raccourcies lorsque le contour est lissé, les sommets peuvent se retrouver au-delà des extrémités d'une crête, ce qui implique que la branche correspondante doit être raccourcie également. Pour ce faire, nous nous basons sur un critère de la méthode DECS (Leborgne et al. 2015), dans laquelle seuls les points $\boldsymbol{\varrho}$ ayant une valeur de crête ($RDG(\boldsymbol{\varrho})$) supérieure au seuil $th_{ridge-low}$ sont conservés comme points de squelette candidats. De la même manière, si la valeur de crête d'un point terminal est en-dessous de $th_{ridge-low}$, nous le supprimons.

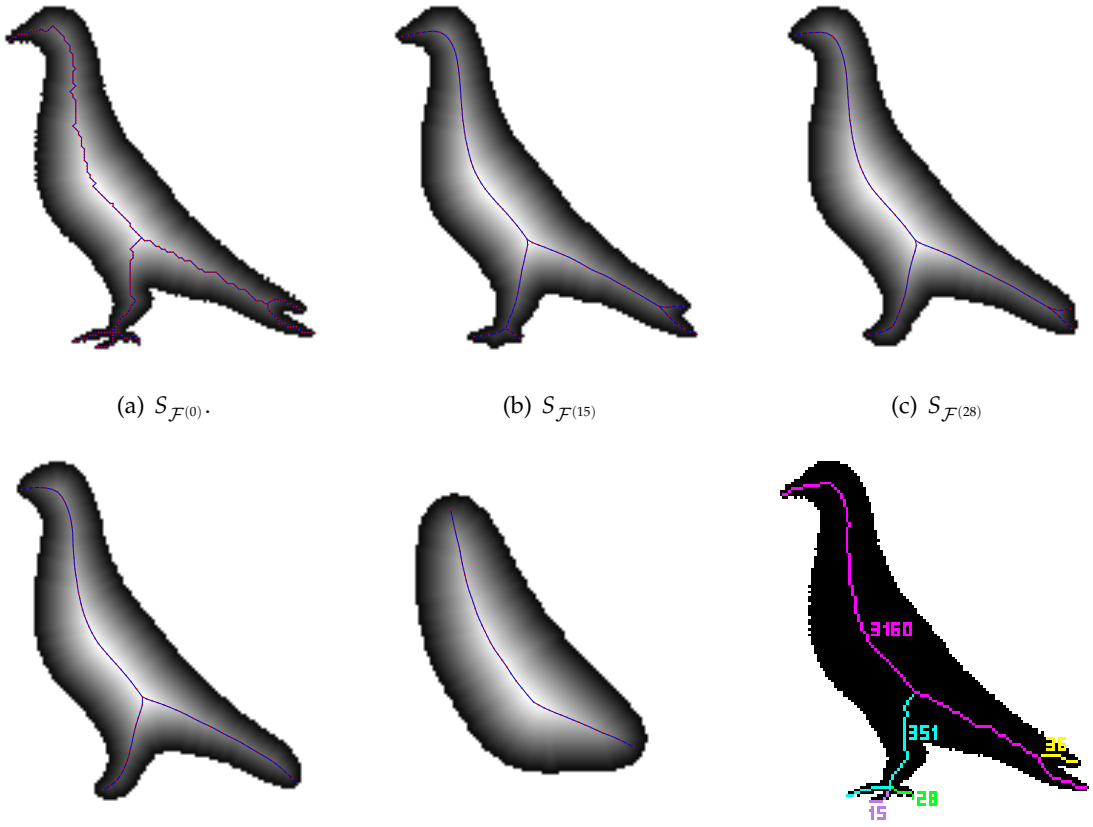
Propagation des étiquettes Lorsqu'un sommet terminal $\mathbf{r}$ est supprimé, soit par ré-échantillonnage soit par élimination des points non-crête, le degré de son sommet adjacent est vérifié. D'une part, s'il est supérieur à 2, la branche se terminant en $\mathbf{r}$ disparaît. D'autre part, si le degré est égal à 2 après suppression de $\mathbf{r}$, les deux branches partant du sommet adjacent n'en forment plus qu'une. Pour permettre la construction du squelette hiérarchique, les étiquettes des branches du squelette initial $S_{\mathcal{F}^{(0)}}$ sont en partie stockées dans les branches du squelette courant. Les arêtes de la même branche devant avoir les mêmes étiquettes, une propagation des étiquettes est effectuée, comme illustré en Figure 4.13. Les étiquettes des deux anciennes branches sont ajoutées aux arêtes de la branche fusionnée. On notera qu'une branche peut contenir un nombre arbitraire d'étiquettes.

Création du squelette hiérarchique (Algorithme 9) Un exemple présentant les étapes critiques lors du processus de déformation du squelette est présenté sur la Figure 4.15. Chaque branche $\flat$ est étiquetée avec une séquence de branches de $S_{\mathcal{F}^{(0)}}$, appelée méta-branche, correspondant à $\flat$ dans $S_{\mathcal{F}^{(0)}}$.

À la fin de cette étape, une séquence de squelettes déformables $h(S)$, comprenant toutes les étapes du squelette déformable calculé sur une forme lissée entre zéro et n_{smooth_max} fois, est obtenue pour représenter l'évolution du squelette d'origine, plus formellement, $h(S) = (S_{\mathcal{F}^{(\lambda)}})_{\lambda \in [0; n_{smooth\_max}]}$. Le squelette hiérarchique S_h est obtenu à partir de l'Algorithme 9. Un exemple est présenté sur la Figure 4.15(f).

Algorithme 8: Pseudo-code du squelette déformable.

Entrée : $S_{\mathcal{F}^{(\lambda)}}$, **Sortie :** $S_{\mathcal{F}^{(\lambda+1)}}$
début
  $S_{\mathcal{F}^{(\lambda+1)}} := S_{\mathcal{F}^{(\lambda)}}$
  tant que $S_{\mathcal{F}^{(\lambda+1)}}$ *n'est pas stable* **faire**
    */*Déformation discrète*/*
    pour tous les *sommets* $\boldsymbol{\varrho}_i$ *de* $S_{\mathcal{F}^{(\lambda+1)}}$ **faire**
      Calculer $\boldsymbol{r}_i$ *et* $\boldsymbol{f}_i$ *selon les équations (4.6) et (4.7)*
      $\boldsymbol{m}_i = \chi \boldsymbol{r}_i + \boldsymbol{f}_i$
    fin
    pour tous les *sommets* $\boldsymbol{\varrho}_i$ *de* $S_{\mathcal{F}^{(\lambda+1)}}$ **faire**
      $\boldsymbol{\varrho}_i = \boldsymbol{\varrho}_i + \Delta t\, \boldsymbol{m}_i$
    fin
    */*Ré-échantillonnage*/*
    pour tous les *sommets* $\boldsymbol{\varrho}_i$*de* $S_{\mathcal{F}^{(\lambda+1)}}$ **faire**
      pour tous les $j \in \eta_i$ **faire**
        si $d(\boldsymbol{\varrho}_i, \boldsymbol{\varrho}_j) > 2c$ **alors**
          Ajouter un sommet entre $\boldsymbol{\varrho}_i$ *et* $\boldsymbol{\varrho}_j$
        sinon si $d(\boldsymbol{\varrho}_i, \boldsymbol{\varrho}_j) < c$ **alors**
          si $|\eta_i| = 1$ **alors**
            Supprimer $\boldsymbol{\varrho}_i$
          sinon si $|\eta_i| = 2$ *et* $|\eta_j| = 2$ **alors**
            Fusionner $\boldsymbol{\varrho}_i$ *et* $\boldsymbol{\varrho}_j$
          sinon si $|\eta_i| = 2$ *et* $|\eta_j| > 2$ **alors**
            Eloigner $\boldsymbol{\varrho}_i$ *à une distance c de* $\boldsymbol{\varrho}_j$
          fin
        fin
      fin
    fin
    pour tous les *sommets terminaux* $\boldsymbol{\varrho}_t$ *de* $S_{\mathcal{F}^{(\lambda+1)}}$ **faire**
      si $|\eta_t| = 1$ *et* $RDG(\boldsymbol{\varrho}_t) < th_{ridge-low}$ **alors**
        Supprimer $\boldsymbol{\varrho}_t$
      fin
    fin
  fin
  retourner $S_{\mathcal{F}^{(\lambda+1)}}$
fin



(a) $S_{\mathcal{F}(0)}$. (b) $S_{\mathcal{F}(15)}$ (c) $S_{\mathcal{F}(28)}$

(d) $S_{\mathcal{F}(36)}$ (e) $S_{\mathcal{F}(351)}$ (f) S_h

FIGURE 4.15 – *Exemple montrant les étapes critiques du squelette déformable et du squelette hiérarchique associé. Les labels des méta-branches sont les nombres de lissages nécessaires pour supprimer les méta-branches.*

Algorithme 9: Pseudo-code de la création de S_h.

Entrées : $h(S)$
Sorties : S_h

```
début
    pour λ allant de n_smooth_max à 0 faire
        pour tous les branches ♭ de S_F(λ) faire
            si ♭.métabranche ∉ S_h alors
                Ajouter ♭.métabranche à S_h
                w_♭.metabranche ← λ
            fin
        fin
    fin
    retourner S_h
fin
```

4.3 MODÉLISATION DU SQUELETTE HIÉRARCHIQUE PAR UN HYPER-GRAPHE HIÉRARCHIQUE

POUR représenter un squelette hiérarchique, nous avons utilisé la structure de données appelée hyper-graphe qui est particulièrement bien adaptée à nos données.

4.3.1 Définition

Un hyper-graphe est un objet mathématique qui généralise la notion de graphe non-orienté. En effet, les arêtes (respectivement, sommets) sont maintenant appelées hyper-arêtes (respectivement, hyper-sommets). Elles relient un nombre d'hyper-sommets compris entre un et le nombre d'hyper-sommets présents dans l'hyper-graphe, alors que dans un graphe non-orienté, une arête ne relie qu'un ou deux sommets au maximum. Grâce à cette notion, nous pouvons définir la notion d'hyper-graphe hiérarchique. Les méta-branches ayant chacune un poids, nous utilisons la notion d'hyper-graphe pondéré pour créer notre hyper-graphe hiérarchique.

Définition 4.3 *Un hyper-graphe hiérarchique H_h est un hyper-graphe pondéré n'ayant aucun sommet isolé. Il est désigné par $H_h = (V, E, w)$, où $V = \{(v_i)\}_{i \in B}$ est un ensemble d'hyper-sommets non vide, $E = (e_j)_{j \in A}$ est un ensemble fini d'hyper-arêtes. Chaque hyper-arête est un sous-ensemble fini de V et $w : V \rightarrow \mathbb{N}$ est le poids quantifiant la hiérarchie. Dans la suite de cette thèse, $w(v_i)$ pourra être noté w_i. B et A sont des ensembles finis d'indices et :*

$$\begin{cases} e_j \neq \emptyset, \forall j \in A \\ \bigcup\limits_{j \in A} e_j = V \end{cases}$$

Habituellement, un hyper-sommet est représenté par un point et une hyper-arête par une courbe fermée ("*patate*") regroupant tous les hyper-sommets qu'elle relie. Un exemple d'hyper-graphe hiérarchique H_h est montré sur la Figure 4.16. Cet hyper-graphe hiérarchique a quatre hyper-sommets pondérés et trois hyper-arêtes. Nous avons :

$$\begin{aligned} &V = \{v_1, v_2, v_3, v_4, v_5\} \\ &E = \{(v_1, v_3, v_4), (v_2, v_5), (v_4, v_5)\} \\ &w(v_1) = 15, w(v_2) = 36, w(v_3) = 28, w(v_4) = 351, w(v_5) = 3160 \end{aligned}$$

Dans notre hyper-graphe hiérarchique, chaque hyper-sommet représente une méta-branche du squelette hiérarchique ayant pour poids celui de la méta-branche représentée. Chaque hyper-arête associe les hyper-sommets ayant un et un seul pixel en commun.

Il convient de justifier cette représentation. Dans les travaux existants, les squelettes sont en général représentés par des graphes dans lesquels les arêtes modélisent les branches du squelette et les sommets, les jonctions entre les branches (Bai et Latecki (2008)). Malheureusement, cette représentation n'est pas possible dans notre cas car une branche hiérarchique est considérée comme la fusion (ou concaténation) de plusieurs branches provenant du squelette

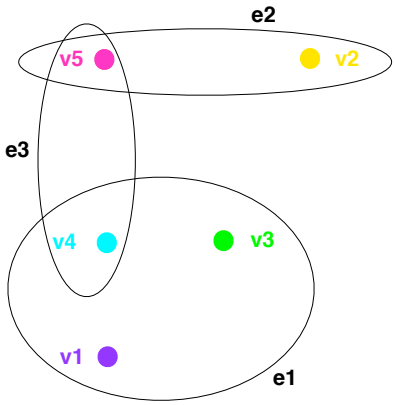


FIGURE 4.16 – *Hyper-graphe hiérarchique construit à partir du squelette hiérarchique présenté sur la Figure 4.15(f).*

d'origine. Les nœuds formés à l'intérieur d'une méta-branche ne peuvent pas être considérés comme des sommets car les méta-branches ne sont pas divisibles. La solution est donc de considérer chaque méta-branche comme un hyper-sommet. De plus, il s'agit d'un bon moyen pour représenter les relations entre les branches lors de l'étape d'appariement (Chapitre 5).

Les hyper-arêtes sont nécessaires pour représenter les adjacences multiples entre les méta-branches (un exemple est l'hyper-arête e_1 sur la Figure 4.16). En effet, dans le squelette hiérarchique, un pixel de jonction peut être partagé par k méta-branches, avec $k \geqslant 1$. Par conséquent, l'utilisation d'hyper-arêtes est absolument nécessaire.

La construction de l'hyper-graphe hiérarchique H_h à partir de S_h (Algorithme 10) est relativement aisée au vu de la structure du squelette hiérarchique. Chaque méta-branche est représentée par un hyper-sommet étiqueté par un poids. Ce poids est normalisé pour qu'il soit compris entre 0 et 1. Ainsi, il est indépendant de la taille de la forme. En effet, plus la forme est grande et plus il faudra d'étapes de lissage pour faire disparaître les branches de squelette. Par ailleurs, chaque relation d'adjacence entre plusieurs méta-branches est une hyper-arête étiquetée par les coordonnées du pixel où se trouve cette relation.

4.4 RÉSULTATS

D'UNE part, nous testons la stabilité sous transformations affines. D'autre part, nous montrons qu'il est possible d'élaguer un squelette à partir de notre hiérarchisation. Finalement, nous étudions la complexité.

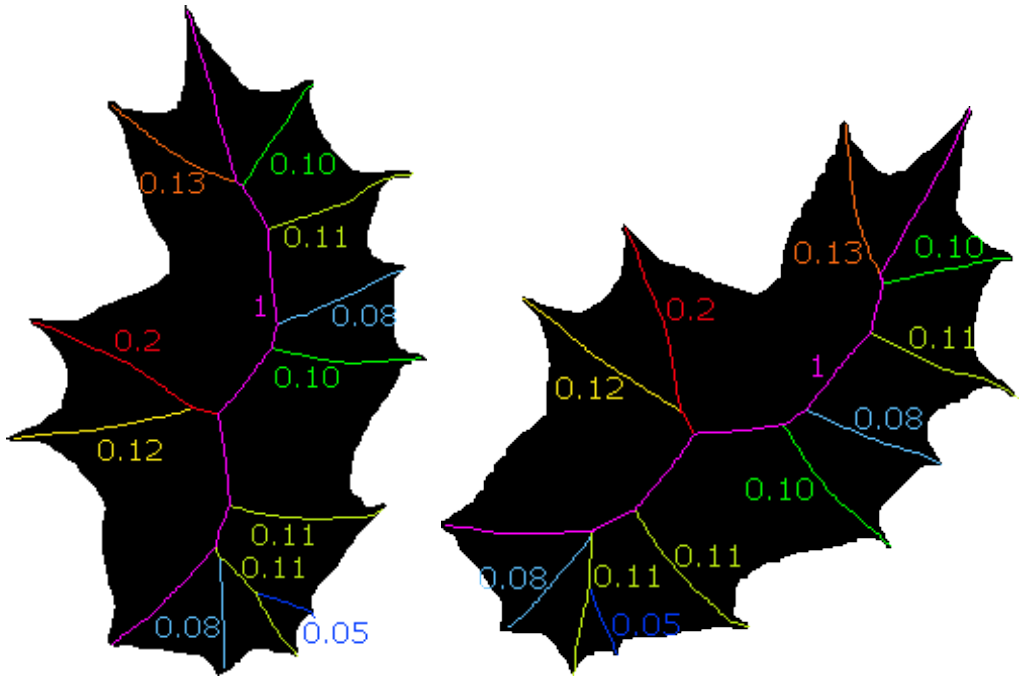
4.4.1 Stabilité sous transformations affines

Nous avons observé que le squelette obtenu avec DECS Leborgne et al. (2015) est stable sous ces transformations. Le squelette déformable est également stable sous rotation. La Figure 4.17 montre deux formes identiques, où la seconde a subi une rotation de 45 degrés. Nous avons testé notre méthode à différentes résolutions sur une même forme. Un exemple est

Algorithme 10: Pseudo-code de la construction d'un hyper-graphe hiérarchique à partir d'un squelette hiérarchique.

Entrées : S_h
Sorties : H_h

```
1  début
2      pour tous les branches ♭ de S_h faire
3          Hyper-sommet nouvelHs← ♭
4          w_nouvelHs ← w_♭ / n_smooth_max
5          Ajouter nouvelHs à H_h
6          pour tous les points p de ♭ faire
7              si B_A(♭, p) > 0 alors
8                  si il n'existe pas une hyper-arête e ayant une étiquette p alors
9                      Hyper-arête nouvelleHa
10                     etiquette(nouvelleHa) ← p
11                     Ajouter ♭ à nouvelleHa
12                     Ajouter nouvelleHa à H_h
13                 sinon
14                     Ajouter ♭ à e
15                 fin
16             fin
17         fin
18     fin
19     retourner H_h
20 fin
```



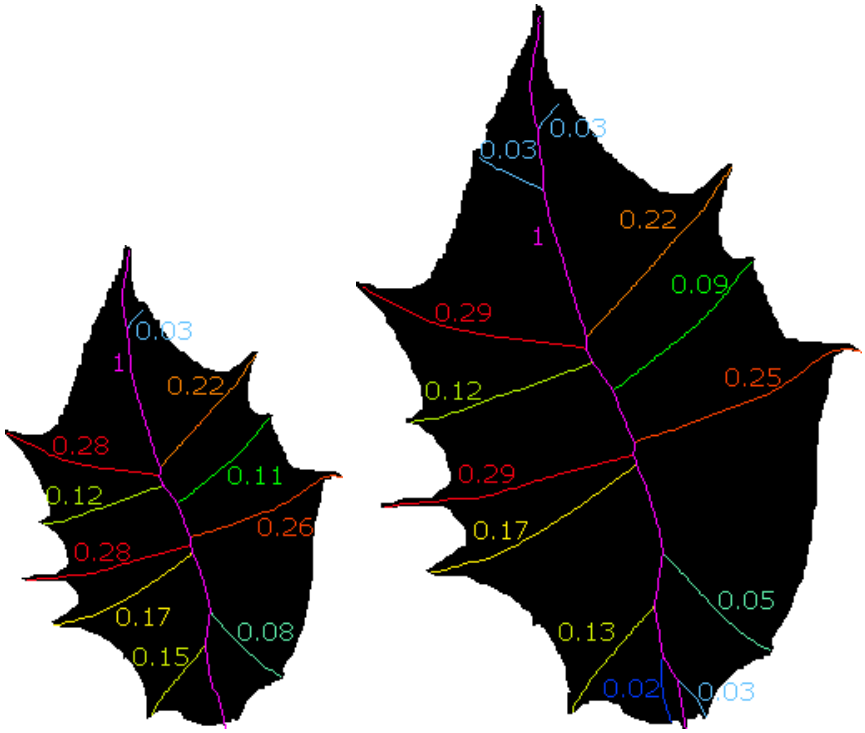
(a) (b)

FIGURE 4.17 – *(a) Squelette hiérarchique d'une feuille de houx, (b) Squelette hiérarchique de la même feuille de houx avec une rotation de 45 degrés.*

montré sur la Figure 4.18. En raison des artéfacts de discrétisation à basse échelle, des détails peuvent être ignorés. Ainsi, le squelette hiérarchique n'est pas nécessairement invariant au changement d'échelle. En effet, lorsque l'échelle augmente, des petits détails du bord peuvent générer des méta-branches supplémentaires non-présentes à basse échelle. Néanmoins, l'intérêt de notre méthode est que ces méta-branches supplémentaires ont un poids minime, proche de zéro. C'est, par exemple, le cas de la branche bleu foncé sur la Figure 4.18(b). Une caractéristique intéressante de notre méthode est que les méta-branches importantes à basse échelle sont les mêmes à plus haute échelle, et leurs poids sont du même ordre de grandeur, comme les branches rouge et fuschia sur la Figure 4.18(b).

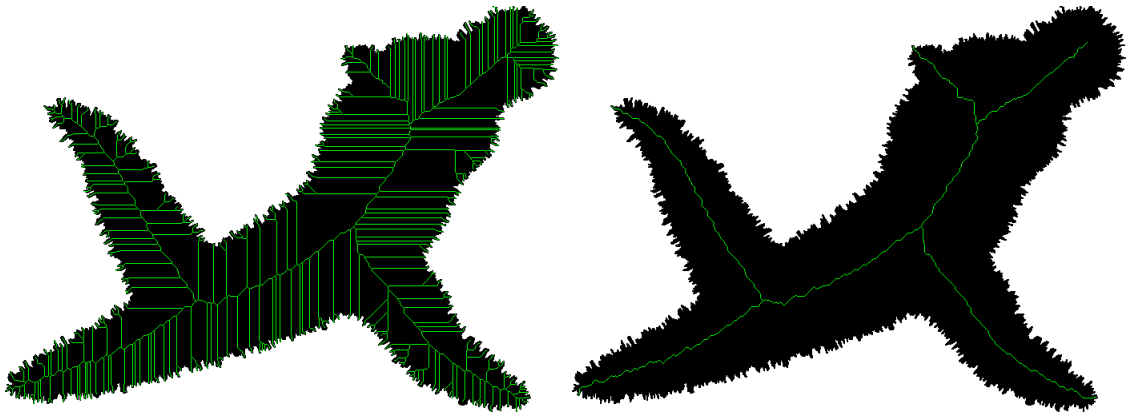
4.4.2 Élagage du squelette

Comme les poids quantifient l'importance des branches, le seuillage par rapport au poids permet d'élaguer naturellement le squelette. Grâce à un seuil $0 \leqslant \sigma \leqslant 1$, nous sélectionnons les méta-branches importantes et obtenons un squelette pertinent. Notre méthode est appliquable sur n'importe quelle autre méthode de squelettisation. En effet, $S_{\mathcal{F}^{(0)}}$ n'est pas nécessairement obtenu grâce à DECS, et peut être remplacé par un squelette issu d'un autre algorithme. La Figure 4.19 montre un exemple d'élagage de squelette obtenu avec l'algorithme de Bertrand et Couprie (Bertrand et Couprie 2014) sur une image très bruitée (nous avons utilisé $\sigma = 0.3$).



(a) (b)

FIGURE 4.18 – *Squelette hiérarchique d'une feuille de houx ayant pour résolution 203 × 291 (a), 304 × 436 (b).*



(a) $S_{\mathcal{F}^{(0)}}$ (b) $S_{0.3}$

FIGURE 4.19 – *Exemple de notre méthode d'élagage appliquée sur un squelette de Bertrand et Couprie (Bertrand et Couprie 2014).*

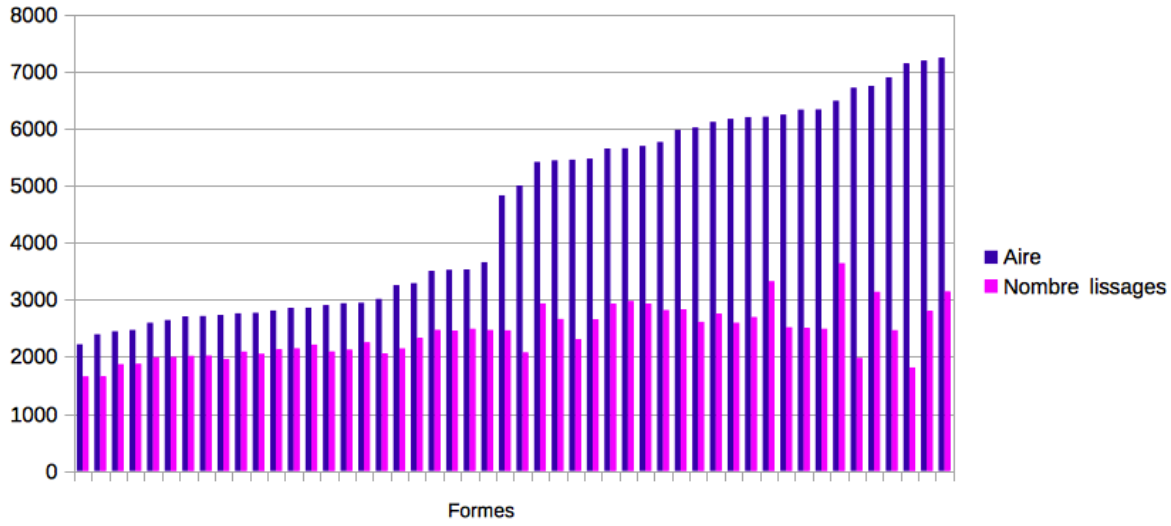


FIGURE 4.20 – *Nombre de lissages nécessaires à la transformation du squelette en un point sur un échantillon de formes ayant chacune une aire différente.*

4.4.3 Complexité

Lors de la hiérarchisation du squelette, l'étape la plus gourmande en temps de calcul est la génération des images lissées jusqu'à l'obtention d'un point en guise de squelette. Nous pouvons majorer cette complexité par $O(n^2)$. Cela signifie que chaque pixel appartenant à $\mathcal{I}$ peut être déplacé du nombre de pixels présents dans l'image. En pratique, la complexité est moindre puisque le nombre de lissages ne dépasse pas l'aire de la forme. En considérant un ensemble de formes triées par aire croissante, la Figure 4.20 montre l'aire et le nombre total d'itérations de lissage n_{smooth_max} pour chaque forme.

La déformation du squelette est une étape qui se déroule en temps linéaire puisqu'il s'agit de parcourir le squelette et de déplacer les points selon le déplacement des crêtes de la carte de distance. La complexité totale en pratique est donc en $O(n^2)$.

CONCLUSION DU CHAPITRE

Dans ce chapitre, nous avons créé un descripteur de formes basé sur le squelette appelé squelette hiérarchique permettant de guider l'algorithme d'appariement. Pour ce faire, nous avons choisi de quantifier l'importance des branches du squelette pour inciter l'algorithme d'appariement à associer les branches ayant le même ordre d'importance entre elles et à donner plus de poids à l'appariement des branches de forte importance, c'est-à-dire celles qui correspondent à l'allure générale de la forme.

Nous avons présenté ainsi une manière d'obtenir un squelette hiérarchique en utilisant une succession de lissages sur la forme initiale pour réduire les détails étape par étape. Puis, nous déformons le squelette au fur et à mesure des lissages de la forme. Comme le lissage réduit les détails de la forme, la longueur des branches du squelette décroît progressivement jusqu'à disparition. Donc, plus une branche disparaît tôt, moins elle a d'importance. Nous avons choisi

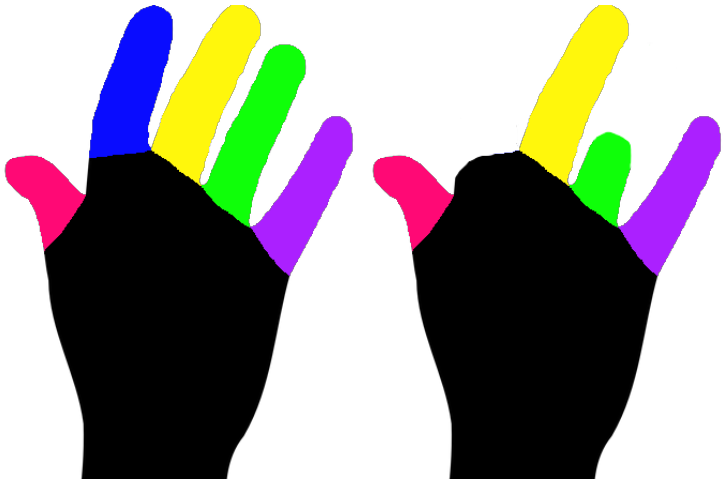
de quantifier l'importance grâce à l'échelle des lissages successifs que nous normalisons pour obtenir une quantification indépendante de la taille des formes. Finalement, nous modélisons le squelette hiérarchique obtenu par un hyper-graphe pondéré afin d'adapter des techniques utilisées en appariement de graphes.

Mis à part le fait que notre squelette hiérarchique soit utilisé en appariement de formes, nous avons également montré qu'il pouvait être utilisé pour réaliser un élagage à partir de n'importe quel type de squelette.

5 APPARIEMENT ET RECONNAISSANCE DE FORMES

SOMMAIRE

5.1 MÉTHODOLOGIE 129
5.1.1 Mesure de dissimilarité entre deux hyper-sommets (méta-branches) provenant de deux hyper-graphes : procédure d'initialisation (1) 129
5.1.2 Appariement des hyper-sommets grâce à l'algorithme hongrois (Kuhn 1955) : procédure d'initialisation (2) 139
5.1.3 Algorithme et exemple de l'étape d'initialisation 140
5.1.4 Raffinement de l'appariement entre hyper-graphes : procédure de mise à jour . . 140
5.1.5 Mesure de dissimilarité globale entre deux hyper-graphes 143
5.2 RÉSULTATS 144
5.2.1 Méthodes utilisées lors de la comparaion 144
5.2.2 Complexité 146
5.2.3 Résultats expérimentaux 148
5.2.4 Application sur une base de données de feuilles d'arbres 155
5.2.5 Exemples d'appariements de feuilles d'arbres 157
CONCLUSION 159



(a) (b)

FIGURE 5.1 – *Illustration de l'appariement des différentes parties d'une main.*

SOUVENONS-NOUS que le but à atteindre, dans cette thèse, est d'identifier une forme inconnue (*forme requête*) en deux dimensions représentée par son squelette hiérarchique, lui-même modélisé par un hyper-graphe hiérarchique. Pour ce faire, nous apparions ce dernier à un ensemble d'hyper-graphes hiérarchiques provenant d'images connues pour déterminer la (ou les) forme(s) qui se rapproche(nt) le plus de la *forme requête*.

Pour apparier deux formes, l'idée est d'associer des parties de formes deux à deux. La Figure 5.1 illustre l'appariement des parties d'une main (celles ayant des couleurs identiques).

L'hyper-graphe hiérarchique extrait au chapitre précédent représente les parties de forme par des hyper-sommets, qui correspondent aux méta-branches du squelette hiérarchique. Nous allons apparier les hyper-sommets provenant de deux hyper-graphes hiérarchiques, en prenant en compte leur importance dans la forme. Une fois le meilleur appariement d'hyper-sommets trouvé, il est quantifié par une mesure de dissimilarité totale. Comme dans le chapitre précédent, notre travail s'applique à des formes sans trou.

Ce chapitre est divisé en deux sections. D'une part, nous exposons la méthodologie (Section 5.1) et d'autre part, nous présentons les résultats (Section 5.2). Le procédé d'appariement se déroule en trois temps. Tout d'abord, nous expliquons le calcul de la mesure de dissimilarité entre deux méta-branches (hyper-sommets) en ne tenant compte que de l'aspect géométrique de la forme reconstruite à partir des méta-branches. Puis, nous utilisons cette mesure de dissimilarité pour en extraire une autre prenant en compte le contexte des méta-branches, c'est-à-dire les relations avec les autres méta-branches du squelette hiérarchique. Cette mesure est notamment basée sur la distance du cantonnier. Enfin, nous appliquons un algorithme d'appariement basé sur la méthode hongroise.

5.1 MÉTHODOLOGIE

L'HYPER-GRAPHE hiérarchique construit dans le chapitre précédent peut également être considéré comme un hyper-graphe relationnel attribué (ARH pour Attributed Relational Hypergraph) puisqu'à chaque hyper-sommet est associée une valeur d'importance (ou attribut) et que chaque hyper-sommet est relié à ses voisins par une hyperarête (relation n-aire). Nous avons donc choisi de baser notre appariement d'hyper-graphes sur une méthode (Kim et al. 2010) utilisant les graphes relationnels attribués (ARG pour Attributed Relational Graph), dans lesquels chaque sommet, que nous pouvons identifier aux hyper-sommets, possède un (ou plusieurs) attribut(s).

L'appariement de deux ARHs consiste à établir une correspondance entre leurs hyper-sommets. La meilleure mise en correspondance/affectation est celle qui minimise le coût total. Le coût d'appariement de deux hyper-sommets tient compte à la fois :

- des attributs/propriétés intrinsèques des hyper-sommets : géométrie des méta-branches, et importance w des méta-branches ;
- du contexte/de la structure local(e) des hyper-sommets, c'est-à-dire des relations d'adjacences contenues dans les hyperarêtes.

Nous proposons d'apparier nos ARHs grâce à une méthode basée sur la structure d'affectation imbriquée (Kim et al. 2010). Cette dernière est implémentée en deux étapes. La première, appelée *étape d'initialisation*, établit une correspondance initiale entre les hyper-sommets pour produire la structure d'affectation imbriquée. Pour ce faire, une matrice de dissimilarité entre hypersommets est construite grâce à l'utilisation de la distance du cantonnier dont les entrées tiennent compte des distances de branches entre les hyper-sommets, du plus court chemin les séparant et de l'importance de chaque hyper-sommet (Section 5.1.1). Puis, un appariement des hyper-sommets est réalisé grâce à l'utilisation de l'algorithme hongrois sur la matrice de dissimilarité entre hyper-sommets précédemment construite (Section 5.1.2). Dans la Section 5.1.3, nous décrivons l'algorithme de l'étape d'initialisation ainsi qu'un exemple. La seconde étape est une *procédure de mise à jour*, qui affine l'appariement initialement trouvé de façon itérative (Section 5.1.4). Finalement, nous extrayons une valeur globale de dissimilarité entre deux hyper-graphes issue de l'appariement trouvé précédemment (Section 5.1.5).

5.1.1 Mesure de dissimilarité entre deux hyper-sommets (méta-branches) provenant de deux hyper-graphes : procédure d'initialisation (1)

Distance de branches entre deux hyper-sommets

Notre méthode, permettant de déterminer une mesure de dissimilarité entre deux hyper-sommets, est inspirée de la distance de chemins (Bai et Latecki (2008), Xu et al. (2009), Shen et al. (2013)) qui obtient de bons résultats en présence d'articulation, d'étirage et de déformation du contour.

Soient $H_h^{(1)}$ (respectivement, $H_h^{(2)}$), l'hyper-graphe hiérarchique associé au squelette hiérarchique $S_h^{(1)}$ (respectivement, $S_h^{(2)}$) provenant de la *forme requête* $\mathcal{F}^{(1)}$ (respectivement, de la

forme connue $\mathcal{F}^{(2)}$). Par abus de notation, une méta-branche sera appelée par l'hyper-sommet la représentant, donc $v_i^{(1)}$ (respectivement, $v_\iota^{(2)}$) est un hyper-sommet de $H_h^{(1)}$ (respectivement, $H_h^{(2)}$). Nous notons $\mid v_i \mid$, le nombre de pixels de la méta-branche v_i.

Définition 5.1 *Soit $v_i = (\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_{|v_i|})$ une méta-branche et $l_E(v_i)$, sa longueur euclidienne définie par :*

$$l_E(v_i) = \sum_{\omega=1}^{|v_i|-1} d_E(\boldsymbol{p}_\omega, \boldsymbol{p}_{\omega+1})$$

où d_E est la distance euclidienne entre deux points.

La première étape consiste à ré-échantillonner les pixels sur chacune des deux méta-branches avec k points pour qu'elles aient le même nombre de points. Rappelons que $edt(\boldsymbol{p})$ est la valeur de la carte de distance euclidienne au point $\boldsymbol{p} \in \mathbb{Z}^2$. La valeur de la carte de distance euclidienne aux points ré-échantillonnés, qui sont à coordonnées réelles, est estimée par interpolation linéaire. On note cette interpolation r. Le ré-échantillonnage des points d'une méta-branche est présenté dans l'Algorithme 11.

La seconde étape est le calcul de la dissimilarité géométrique à partir du ré-échantillonnage. La dissimilarité est fonction de la somme des différences entre les rayons des boules, dont les centres sont les points échantillonnés, mis en correspondance pour avoir une certaine invariance aux déformations articulées. Cela a été proposé par Bai et Latecki (2008). Autrement dit, le rayon de la boule du $\omega^{ème}$ point échantillonné de $\tilde{v}_i^{(1)}$ est comparé au $\omega^{ème}$ (en violet sur la Figure 5.2) et au $(\mid \tilde{v}_\iota^{(2)} \mid -\omega)^{ème}$ (en vert sur la Figure 5.2) points échantillonnés de $\tilde{v}_\iota^{(2)}$. Nous prenons également en compte la différence de longueur entre deux méta-branches.

Pour rendre cette mesure de dissimilarité invariante à la taille de la forme, nous utilisons le facteur de normalisation suivant :

$$\aleph(\mathcal{F}) = 1 + \frac{1}{\frac{1}{\mid \mathcal{F} \mid} \sum_{\boldsymbol{p} \in \mathcal{F}} edt(\boldsymbol{p})} \tag{5.1}$$

Nous notons $\xi l(v_i^{(1)}, v_\iota^{(2)})$, la différence de longueur normalisée entre deux méta-branches. Elle est définie par :

$$\xi l(v_i^{(1)}, v_\iota^{(2)}) = |l_E(v_i^{(1)})\aleph(\mathcal{F}^{(1)}) - l_E(v_\iota^{(2)})\aleph(\mathcal{F}^{(2)})| \tag{5.2}$$

Nous notons $\xi r(\tilde{\boldsymbol{p}}_\omega^{(1)}, \tilde{\boldsymbol{p}}_{\omega'}^{(2)})$, la différence de rayon normalisée au carré entre deux points provenant respectivement des méta-branches échantillonnées $\tilde{v}_i^{(1)}$ et $\tilde{v}_\iota^{(2)}$. Elle est définie par :

$$\xi r(\tilde{\boldsymbol{p}}_\omega^{(1)}, \tilde{\boldsymbol{p}}_{\omega'}^{(2)}) = (r(\tilde{\boldsymbol{p}}_\omega^{(1)})\aleph(\mathcal{F}^{(1)}) - r(\tilde{\boldsymbol{p}}_{\omega'}^{(2)})\aleph(\mathcal{F}^{(2)}))^2 \tag{5.3}$$

La dissimilarité entre deux méta-branches, appelée *distance de branches* et notée db, est calculée comme la somme pondérée du terme de différence de rayons et du terme de longueur.

Algorithme 11: Ré-échantillonnage d'une méta-branche avec k points.

Entrées : k : nombre de points à échantillonner, v_i une méta-branche, EDT : Transformée en Distance Euclidienne

Sorties : $\tilde{v}_i$: la méta-branche v_i ré-échantillonnée avec k points

Variable : $\boldsymbol{q}$: point courant appartenant à $\mathbb{Z}^2$.

début
- $\boldsymbol{q} \leftarrow \boldsymbol{p}_1$;
- $l \leftarrow 0$;
- $\omega \leftarrow 2$;
- **tant que** $\omega <$ | v_i | **faire**
  - **si** $d_E(\boldsymbol{q}, \boldsymbol{p}_\omega) + l < \frac{l_E(v_i)}{k-1}$ **alors**
    - $l \leftarrow l + d_E(\boldsymbol{q}, \boldsymbol{p}_\omega)$;
    - $\boldsymbol{q} \leftarrow \boldsymbol{p}_\omega$;
    - $\omega + +$;
  - **sinon**
    - **si** $d_E(\boldsymbol{q}, \boldsymbol{p}_\omega) + l = \frac{l_E(v_i)}{k-1}$ **alors**
      - $\boldsymbol{q} \leftarrow \boldsymbol{p}_\omega$;
      - $l \leftarrow 0$;
      - $r(\boldsymbol{q}) \leftarrow edt(\boldsymbol{q})$;
      - *Ajouter* $\boldsymbol{q}$ *à* $\tilde{v}_i$;
      - $\omega + +$;
    - **sinon**
      - $r(\boldsymbol{q}) \leftarrow \frac{l}{d_E(\boldsymbol{q}, \boldsymbol{p}_\omega)} r(\boldsymbol{q}) + (1 - \frac{l}{d_E(\boldsymbol{q}, \boldsymbol{p}_\omega)}) edt(\boldsymbol{p}_\omega)$;
      - $\boldsymbol{q} \leftarrow \boldsymbol{q} + (\frac{l_E(v_i)}{k-1} - l)\overrightarrow{\boldsymbol{q}\boldsymbol{p}_\omega}$;
      - $l \leftarrow 0$;
      - *Ajouter* $\boldsymbol{q}$ *à* $\tilde{v}_i$;
    - **fin**
  - **fin**
- **fin**
- **retourner** $\tilde{v}_i$;

fin

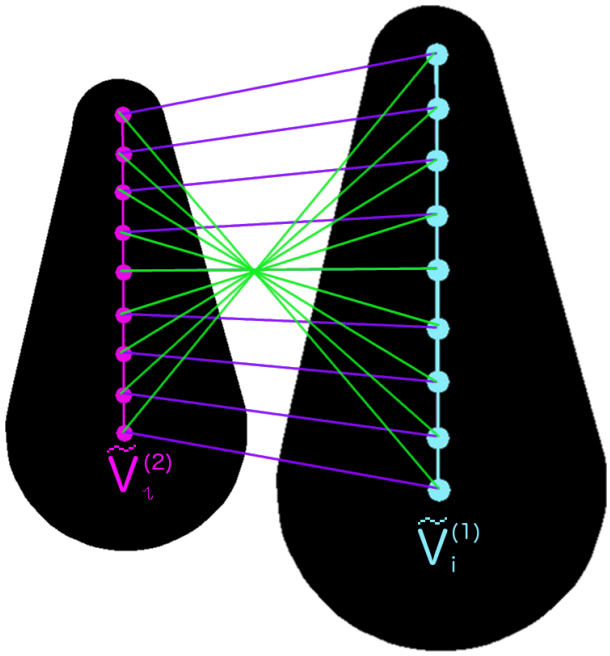


FIGURE 5.2 – *Illustration de l'échantillonnage de deux hyper-sommets ainsi que des deux mises en correspondance possibles des points échantillonnés.*

La somme des différences entre rayons est calculée pour les deux sens de parcours, la plus avantageuse étant conservée :

$$db(v_i^{(1)}, v_\iota^{(2)}) = \alpha \min\left(\sum_{\omega=1}^{k} \xi r(\tilde{\boldsymbol{p}}_\omega^{(1)}, \tilde{\boldsymbol{p}}_\omega^{(2)}), \sum_{\omega=1}^{k} \xi r(\tilde{\boldsymbol{p}}_\omega^{(1)}, \tilde{\boldsymbol{p}}_{k-\omega+1}^{(2)})\right) + (1-\alpha)\xi l(v_i^{(1)}, v_\iota^{(2)}) \quad (5.4)$$

Les valeurs du nombre de points échantillonnés par branche, k, et du poids $\alpha \in [0,1]$ sont discutées dans la Section 5.2.

Plus court chemin entre deux méta-branches d'un même hyper-graphe

Pour intégrer le contexte dans le coût d'appariement entre deux méta-branches $v_i^{(1)}$ et $v_\iota^{(2)}$, nous allons associer $v_i^{(1)}$ (respectivement, $v_\iota^{(2)}$) à chacune des autres méta-branches de $H_h^{(1)}$ (respectivement, $H_h^{(2)}$). Nous notons $v_i^{(1)} \smile v_{i'}^{(1)}$ (respectivement, $v_\iota^{(2)} \smile v_{\iota'}^{(2)}$, l'association des méta-branches i et i' (respectivement, ι et ι') dans $H_h^{(1)}$ (respectivement, $H_h^{(2)}$). Pour ce faire, nous avons choisi d'exploiter la "position relative" d'une méta-branche par rapport à toutes les autres. Cette position relative est extraite en calculant le plus court chemin reliant $v_i^{(1)}$ (respectivement, $v_\iota^{(2)}$) à toutes les méta-branches de $H_h^{(1)}$ (respectivement, $H_h^{(2)}$) en passant uniquement par les méta-branches de $H_h^{(1)}$ (respectivement, $H_h^{(2)}$). Plus précisément, une hyper-arête correspondant à un unique point de jonction entre deux méta-branches ou plus, le plus court chemin entre deux méta-branches est le plus court chemin entre deux points de jonction, en suivant les pixels de squelette. Ainsi, une association entre deux méta-branches comprend ces deux dernières et le plus court chemin les reliant. Sur la Figure 5.3, la branche fuschia est associée à la blanche en passant par le plus court chemin en violet.

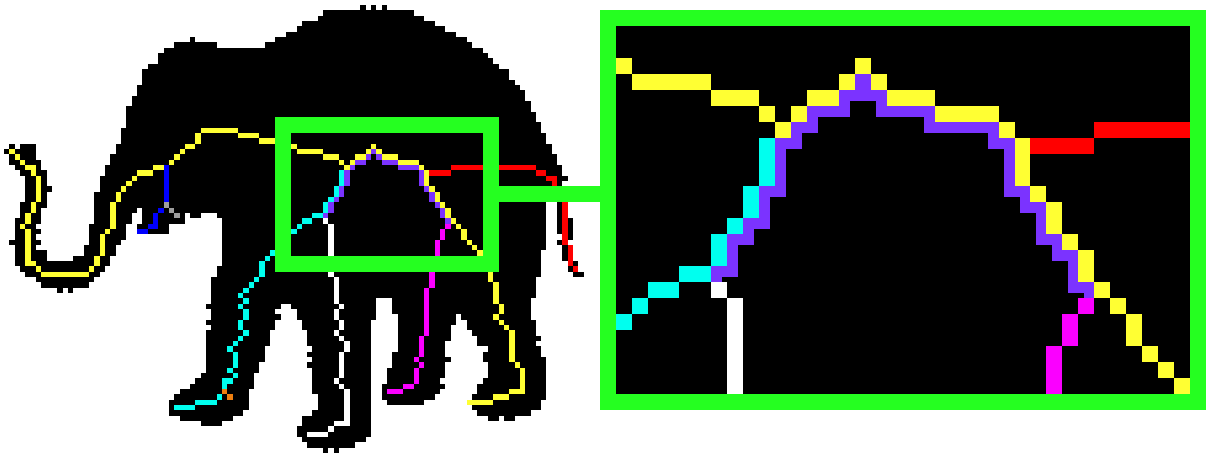


FIGURE 5.3 – *Exemple du plus court chemin entre deux méta-branches.*

Dans le cadre d'un arbre (nous travaillons avec des formes sans trou), il n'existe qu'un seul chemin entre deux hyper-arêtes données. Néanmoins, un de nos objectifs étant d'étendre notre travail aux formes trouées, il est nécessaire d'avoir un algorithme générique. Prenons l'exemple de la Figure 5.3 dans laquelle le plus court chemin entre la méta-branche fuchsia et la blanche est représenté en violet. Il passe par les méta-branches jaune et turquoise.

Pour ce faire, commençons par calculer le plus court chemin entre deux hyper-arêtes e_s (source) et e_d (destination), grâce à l'Algorithme de Dijkstra (Dijkstra (1959), Chen (2003)) appliqué aux hyper-graphes (*cf.* Algorithme 12). Dijkstra utilise le fait que si e_j est une hyper-arête sur le plus court chemin entre e_s et e_d, la connaissance de ce dernier implique la connaissance du plus court chemin entre e_s et e_j. Les plus courts chemins à partir de e_j vers les autres hyper-arêtes sont construits pour accroître la longueur jusqu'à atteindre e_d.

Pour comprendre cet algorithme, il est nécessaire de définir deux notions.

Définition 5.2 *Le **voisinage d'une hyper-arête** e_j, noté $\mathcal{V}(e_j)$, appartenant à E dans $H_h = (V, E, w)$ est défini par :*

$$\mathcal{V}(e_j) = \{e_{j'} \mid \exists v_i \in V \mid v_i \subset e_j \;\&\; v_i \subset e_{j'}\}$$

Définition 5.3 *Soit l'hyper-sommet $v_i = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{|v_i|})$ appartenant à e_j et à $e_{j'}$, et $etiquette(e_j) = \boldsymbol{p}_\mu$, $etiquette(e_{j'}) = \boldsymbol{p}_{\mu'}$ appartenant à v_i tel que $1 \leq \mu \leq \mu' \leq |v_i|$. La **distance euclidienne entre deux hyper-arêtes voisines** e_j et $e_{j'}$, en ne parcourant que les pixels de v_i compris entre $\boldsymbol{p}_\mu$ et $\boldsymbol{p}_{\mu'}$, est définie par :*

$$\widehat{d_E}(e_j, e_{j'}) = \widehat{d_E}(e_{j'}, e_j) = \sum_{\omega=\mu}^{\mu'-1} d_E(\boldsymbol{p}_\omega, \boldsymbol{p}_{\omega+1})$$

Pour déterminer le chemin le plus court entre deux méta-branches (ou hyper-sommets) v_s (source) et v_d (destination), il est nécessaire de calculer le plus court chemin entre chaque hyper-arête incluant v_s et chaque hyper-arête incluant v_d pour ne récupérer que la valeur minimale (*cf.* Algorithme 13). De cette manière, la longueur de v_s et v_d n'est pas prise en compte dans le calcul du plus court chemin entre ces deux hyper-sommets. Nous notons $\breve{\mathrm{d}}_E(v_s, v_d)$ la longueur de ce chemin.

Algorithme 12: Calcul du plus court chemin entre deux hyper-arêtes grâce à l'Algorithme de Dijkstra (Dijkstra (1959), Chen (2003)).

Entrées : H_h, e_s (hyper-arête source) et e_d (hyper-arête destination)
Sorties : $l(e_d)$ longueur du plus court chemin entre e_s et e_d

fonction : longueurPlusCourtCheminEntre2Hyperaretes(e_s, e_d)**{**
pour tous les *hyper-arêtes e appartenant à* H_h **faire**
  $marquage(e) \leftarrow faux$
  $l(e) \leftarrow +\infty$
fin
$l(e_s) \leftarrow 0$
tant que $marquage(e_d) = faux$ **faire**
  Choisir l'hyper-arête non marquée e pour laquelle $l(e)$ *est minimale*
  $marquage(e) \leftarrow vrai$
  pour tous les $e' \in \mathcal{V}(e)$ *t.q.* $marquage(e') = faux$ **faire**
    $l(e') = \min(l(e'), l(e) + \widehat{\mathrm{d}}_E(e, e'))$
  fin
fin
retourner $l(e_d)$ **}**

Algorithme 13: Calcul de la longueur du plus court chemin entre deux hyper-sommets en utilisant l'Algorithme de Dijkstra entre hyper-arêtes.

Entrées : H_h, v_s (hyper-sommet source) et v_d (hyper-sommet destination)
Sorties : l longueur du plus court chemin entre v_s et v_d

fonction : longueurPlusCourtCheminEntre2Hypersommets(e_s, e_d)**{**
$l \leftarrow +\infty$
pour tous les *hyper-aretes e incluant* v_s **faire**
  pour tous les *hyper-aretes* e' *incluant* v_d **faire**
    si $l > longueurPlusCourtChemin(e, e')$ **alors**
      $l \leftarrow longueurPlusCourtCheminEntre2Hyperaretes(e, e')$
    fin
  fin
fin
retourner l **}**

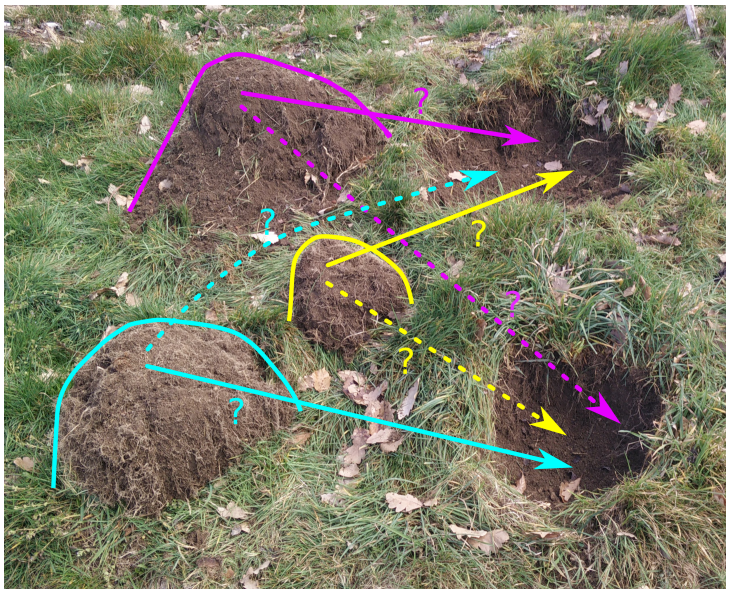


FIGURE 5.4 – *Illustration de l'EMD grâce à l'image des tas de terre et des trous. Les flèches pleines représentent le déplacement optimal des tas de terre vers les trous.*

Coût d'appariement entre deux hyper-sommets

Pour un couple d'hyper-sommets $(v_i^{(1)}, v_l^{(2)})$ donné, nous avons $|V^{(1)}||V^{(2)}|$ dissimilarités entre associations. Or, nous ne devons obtenir qu'une seule valeur représentant l'appariement entre deux hyper-sommets.

Cet appariement est calculé à l'aide de la distance du cantonnier (EMD, pour Earth Mover's Distance). Au cours de cette sous-section, nous nous référerons à l'image des tas de terre décrite ci-après afin de bien comprendre le mécanisme qui sous-tend le calcul de cette distance tout en faisant le parallèle avec les données que nous manipulons.

Considérons deux distributions. La première est vue comme un ensemble de tas de terre ayant chacun un certain volume dans un espace, la seconde comme un ensemble de trous de volume plus ou moins important dans le même espace. L'EMD permet de calculer, de façon imagée, la plus petite quantité de travail à fournir pour boucher les trous avec l'ensemble des tas de terre en tenant compte du volume de terre et de la distance à parcourir entre l'endroit où la terre est prise et le trou dans lequel elle est mise (dans le cadre de l'EMD, nous parlons de **distance au sol**). Une illustration est proposée sur la Figure 5.4. Le tas de terre jaune étant le plus petit et le plus près des trous, c'est celui qui demandera une moins grande quantité de travail pour le déplacer dans un des deux trous. Par ailleurs, sur cet exemple, la quantité minimale de travail est obtenue en déplaçant les tas fuschia et jaune dans le trou du haut et le tas turquoise dans le trou du bas. Il est à noter que si le trou du bas est bouché avec le tas turquoise ayant le même volume que le trou, il est impossible de mettre le tas jaune dans ce même trou puisque son volume maximal est atteint.

Dans notre cas, nous utilisons l'EMD pour déterminer une valeur de dissimilarité entre deux hyper-sommets $v_i^{(1)}$ et $v_l^{(2)}$. Elle permet de calculer la quantité de travail à effectuer pour obtenir $v_l^{(2)}$ à partir de $v_i^{(1)}$ en tenant compte du contexte de ces hyper-sommets, c'est-à-dire,

d'une part, le plus court chemin reliant $v_i^{(1)}$ (respectivement, $v_\iota^{(2)}$) aux hyper-sommets appartenant au même hyper-graphe, d'autre part, la distance de branche entre $v_i^{(1)}$ et $v_\iota^{(2)}$ et celle entre chaque paire d'hyper-sommets provenant des deux hyper-graphes.

Pour faire l'analogie avec les tas de terre et les trous, un tas de terre représente $v_i^{(1)}$ associé à un autre hyper-sommet $v_{i'}^{(1)}$ de $H_h^{(1)}$ ($v_i^{(1)} \smile v_{i'}^{(1)}$). Un trou représente $v_\iota^{(2)}$ associé à un autre hyper-sommet $v_{\iota'}^{(2)}$ de $H_h^{(2)}$ ($v_\iota^{(2)} \smile v_{\iota'}^{(2)}$). Le volume des tas de terre (respectivement, des trous) est représenté par l'importance des hyper-sommets, auxquels est associé $v_i^{(1)}$ (respectivement, $v_\iota^{(2)}$), déterminée grâce à la hiérarchisation des méta-branches présentée au Chapitre 4.

La solution de ce problème peut être modélisée par le problème d'optimisation linéaire suivant. Soit

$$\varpi_i^{(1)} = \{(v_i^{(1)} \smile v_0^{(1)}, w(v_0^{(1)})), \ldots, (v_i^{(1)} \smile v_{|V^{(1)}|}^{(1)}, w(v_{|V^{(1)}|}^{(1)}))\},$$

la distribution discrète caractérisant $v_i^{(1)}$ et

$$\varpi_\iota^{(2)} = \{(v_\iota^{(2)} \smile v_0^{(2)}, w(v_0^{(2)})), \ldots, (v_\iota^{(2)} \smile v_{|V^{(2)}|}^{(2)}, w(v_{|V^{(2)}|}^{(2)}))\},$$

la distribution discrète caractérisant $v_\iota^{(2)}$ où $v_i^{(\tau)} \smile v_{i'}^{(\tau)}$ est la notation de l'association de $v_i^{(\tau)}$ et $v_{i'}^{(\tau)}$ dans l'hyper-graphe $H_h^{(\tau)}$, comme vu précédemment. $w(v_{i'}^{(\tau)})$ représente l'importance de $v_{i'}^{(\tau)}$ dans l'hyper-graphe $H_h^{(\tau)}$ et $|V^{(\tau)}|$, le nombre d'hyper-sommets dans l'hyper-graphe $H_h^{(\tau)}$.

Définition 5.4 *La dissimilarité entre $v_i^{(1)} \smile v_{i'}^{(1)}$ et $v_\iota^{(2)} \smile v_{\iota'}^{(2)}$ est définie comme suit :*

$$\psi\left(v_i^{(1)} \smile v_{i'}^{(1)}, v_\iota^{(2)} \smile v_{\iota'}^{(2)}\right) =$$
$$\beta\left(db(v_i^{(1)}, v_\iota^{(2)}) + db(v_{i'}^{(1)}, v_{\iota'}^{(2)})\right) + (1-\beta)|\breve{d}_E(v_i^{(1)}, v_{i'}^{(1)}) - \breve{d}_E(v_\iota^{(2)}, v_{\iota'}^{(2)})|$$

où db note la distance de branches entre deux hyper-sommets (Équation 5.4) et $\breve{d}_E$ la longueur du plus court chemin entre deux hyper-sommets (Algorithme 13).

Lorsqu'on fait l'analogie entre les tas et les trous, la dissimilarité précédemment énoncée représente la distance au sol entre le tas de terre correspondant à l'association $v_i^{(1)} \smile v_{i'}^{(1)}$ et le trou correspondant à l'association $v_\iota^{(2)} \smile v_{\iota'}^{(2)}$.

Nous notons $\boldsymbol{\psi}\left(v_i^{(1)}, v_\iota^{(2)}\right) = \left[\psi\left(v_i^{(1)} \smile v_{i'}^{(1)}, v_\iota^{(2)} \smile v_{\iota'}^{(2)}\right)\right]_{\substack{0 \le i' \le |V^{(1)}| \\ 0 \le \iota' \le |V^{(2)}|}}$, la matrice regroupant toutes les mesures de dissimilarité entre chaque élément de $\varpi_i^{(1)}$ (tas de terre) et chaque élément de $\varpi_\iota^{(2)}$ (trou).

Nous définissons $\boldsymbol{\psi}$ par :

$$\boldsymbol{\psi} = \left[\boldsymbol{\psi}(v_i^{(1)}, v_\iota^{(2)})\right]_{\substack{0 \le i \le |V^{(1)}| \\ 0 \le \iota \le |V^{(2)}|}}$$

Pour déterminer la quantité de travail minimale à réaliser pour transporter la terre provenant des tas vers les trous, il est nécessaire de déterminer un flot $\beth$ tel que :

$$\beth_{\varpi_\iota^{(2)}}^{\varpi_i^{(1)}} = \left[\zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}}\right]_{\substack{0 \le i' \le |V^{(1)}| \\ 0 \le \iota' \le |V^{(2)}|}}$$

où $\zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}}$ est le flot entre $v_i^{(1)} \smile v_{i'}^{(1)}$ et $v_\iota^{(2)} \smile v_{\iota'}^{(2)}$ qui minimise le coût total suivant :

$$TRAVAIL(\omega_i^{(1)}, \omega_\iota^{(2)}, \exists_{\omega_\iota^{(2)}}^{\omega_i^{(1)}}) = \sum_{i'=1}^{|V^{(1)}|} \sum_{\iota'=1}^{|V^{(2)}|} \psi\left(v_i^{(1)} \smile v_{i'}^{(1)}, v_\iota^{(2)} \smile v_{\iota'}^{(2)}\right) \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}}$$

en vérifiant les quatre contraintes suivantes :

$$\zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}} \geq 0 \qquad 1 \leq i' \leq |V^{(1)}|, 1 \leq \iota' \leq |V^{(2)}| \tag{5.5}$$

$$\sum_{\iota'=1}^{|V^{(2)}|} \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}} \leq w(v_{i'}^{(1)}) \qquad 1 \leq i' \leq |V^{(1)}| \tag{5.6}$$

$$\sum_{i'=1}^{|V^{(1)}|} \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}} \leq w(v_{\iota'}^{(2)}) \qquad 1 \leq \iota' \leq |V^{(2)}| \tag{5.7}$$

$$\sum_{i'=1}^{|V^{(1)}|} \sum_{\iota'=1}^{|V^{(2)}|} \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}} = \min\left(\sum_{i'=1}^{|V^{(1)}|} w(v_{i'}^{(1)}), \sum_{\iota'=1}^{|V^{(2)}|} w(v_{\iota'}^{(2)})\right) \tag{5.8}$$

La contrainte (5.5) permet de déplacer la terre provenant des tas vers les trous et non l'inverse. La contrainte (5.6) limite le volume de terre qui peut être pris dans chaque tas au volume de ces derniers. La contrainte (5.7) limite le volume de terre qui peut-être reçu par chaque trou au volume de ces derniers. Finalement, la contrainte (5.8) indique que la quantité de terre déplacée sera minimale entre le volume total des tas et le volume total des trous.

Le calcul de l'EMD est donc un problème de programmation linéaire qui consiste à minimiser un coût tout en respectant un ensemble de contraintes. Il est résolu par l'algorithme du Simplexe (Hillier et Lieberman (1995)). Cette méthode se déroule en deux phases :

1. Initialisation : trouver une solution de base réalisable ;
2. Progression : itérativement, améliorer la résolution de la fonction objectif. Le processus se termine lorsque la fonction objectif n'est plus améliorable ;

Une fois ce problème de programmation linéaire résolu, le flot optimal $\exists_{\omega_\iota^{(2)}}^{\omega_i^{(1)}}$ est déterminé. La distance du cantonnier est alors définie par :

$$emd(\omega_i^{(1)}, \omega_\iota^{(2)}) = \frac{\sum_{i'=1}^{|V^{(1)}|} \sum_{\iota'=1}^{|V^{(2)}|} \psi\left(v_i^{(1)} \smile v_{i'}^{(1)}, v_\iota^{(2)} \smile v_{\iota'}^{(2)}\right) \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}}}{\sum_{i'=1}^{|V^{(1)}|} \sum_{\iota'=1}^{|V^{(2)}|} \zeta_{v_\iota^{(2)} \smile v_{\iota'}^{(2)}}^{v_i^{(1)} \smile v_{i'}^{(1)}}} \tag{5.9}$$

Elle représente la dissimilarité brute entre les méta-branches $v_i^{(1)}$ et $v_\iota^{(2)}$.

Nous avons choisi d'utiliser l'EMD pour plusieurs raisons. Cette distance permet :

– une grande flexibilité concernant la nature des mesures à prendre en compte, ce qui nous a permis de l'appliquer facilement à nos données ;
– de prendre en considération à la fois la dissimilarité entre les branches et leur importance ;

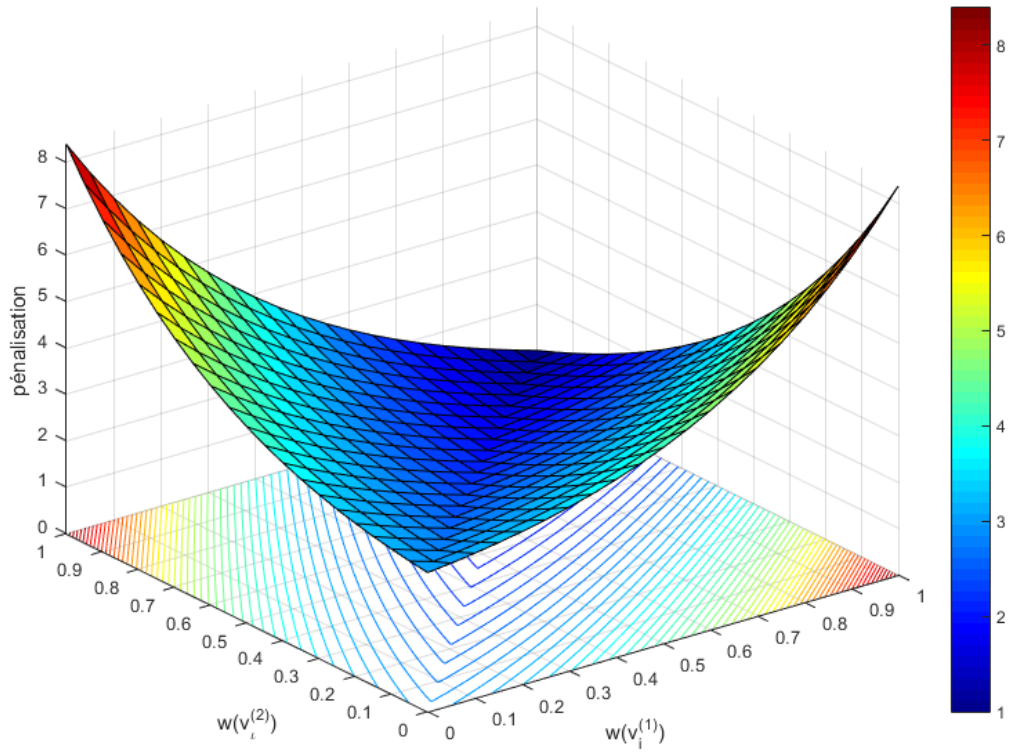


FIGURE 5.5 – *Fonction de pénalisation basée sur l'importance des hyper-sommets.*

- les appariements partiels de façon très naturelle, ce qui s'avère très utile lorsque le squelette est bruité ou qu'il y a une occultation de la forme ;
- un nombre différent d'éléments dans les deux distributions. Il est donc possible d'obtenir une mesure de dissimilarité basée sur un appariement entre hyper-sommets même si les deux hyper-graphes hiérarchiques n'ont pas le même nombre d'hyper-sommets.

Par ailleurs, intuitivement, les branches ayant un degré d'importance proche doivent en priorité s'apparier. Parmi celles qui ont la même importance, nous voulons favoriser celles pour lesquelles l'importance est élevée. En d'autres termes, nous voulons comparer absolument l'allure générale de deux formes puis les détails ayant le même ordre d'importance. Pour ce faire, nous avons déterminé une fonction permettant de pénaliser fortement la différence d'importance des branches, d'où l'utilisation de la fonction exponentielle dans l'Équation 5.10. Dans une moindre mesure, nous pénalisons la valeur de l'importance des branches grâce au dernier terme de l'Équation 5.10. Le coût d'appariement est donc déterminé par :

$$\Xi\left(v_i^{(1)}, v_l^{(2)}\right) = emd(\omega_i^{(1)}, \omega_l^{(2)}) \left(e^{2|w(v_i^{(1)}) - w(v_l^{(2)})|} + (2 - w(v_i^{(1)}) - w(v_l^{(2)}))\right) \tag{5.10}$$

La Figure 5.5 représente la fonction de pénalisation pour les valeurs d'importance des hyper-sommets de $H_h^{(1)}$ et $H_h^{(2)}$ comprises entre zéro et 1 ainsi que ses lignes de niveau.

$\Xi\left(v_1^{(1)}, v_1^{(2)}\right)$	$\Xi\left(v_1^{(1)}, v_2^{(2)}\right)$	...	$\Xi\left(v_1^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_1^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$
$\Xi\left(v_2^{(1)}, v_1^{(2)}\right)$	$\Xi\left(v_2^{(1)}, v_2^{(2)}\right)$	...	$\Xi\left(v_2^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_2^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$
...	...	...	...	...	...
$\Xi\left(v_i^{(1)}, v_1^{(2)}\right)$	$\Xi\left(v_i^{(1)}, v_2^{(2)}\right)$	...	$\Xi\left(v_i^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_i^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$
...	...	...	...	...	...
$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_1^{(2)}\right)$	$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_2^{(2)}\right)$	...	$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$

TABLE 5.1 – *Construction de la matrice* Ξ

$\Xi\left(v_1^{(1)}, v_1^{(2)}\right)$	...	$\Xi\left(v_1^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_1^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$	$moy(\Xi)$	$moy(\Xi)$
$\Xi\left(v_2^{(1)}, v_1^{(2)}\right)$	...	$\Xi\left(v_2^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_2^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$	$moy(\Xi)$	$moy(\Xi)$
...	...	...	...	...	$moy(\Xi)$	$moy(\Xi)$
$\Xi\left(v_i^{(1)}, v_1^{(2)}\right)$	...	$\Xi\left(v_i^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_i^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$	$moy(\Xi)$	$moy(\Xi)$
...	...	...	...	...	$moy(\Xi)$	$moy(\Xi)$
$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_1^{(2)}\right)$	...	$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_\iota^{(2)}\right)$	...	$\Xi\left(v_{\lvert V^{(1)}\rvert}^{(1)}, v_{\lvert V^{(2)}\rvert}^{(2)}\right)$	$moy(\Xi)$	$moy(\Xi)$

TABLE 5.2 – *Construction de la matrice* $\Xi^{\square}$ *à partir de* Ξ *lorsque* $|V^{(1)}| = |V^{(2)}| + 2 = |V_{max}|$.

5.1.2 Appariement des hyper-sommets grâce à l'algorithme hongrois (Kuhn 1955) : procédure d'initialisation (2)

Dans la partie précédente, nous avons vu comment déterminer le coût d'appariement entre deux hyper-sommets $v_i^{(1)}$ et $v_\iota^{(2)}$. Par conséquent, dans cette partie, nous considérons que nous connaissons le coût d'appariement entre chaque hyper-sommet de $H_h^{(1)}$ et chaque hyper-sommet de $H_h^{(2)}$. Nous attribuons chaque ligne (respectivement, colonne) à un hyper-sommet de $H_h^{(1)}$ (respectivement, $H_h^{(2)}$) d'une matrice appelée Ξ. Ainsi, $\Xi\left(v_i^{(1)}, v_\iota^{(2)}\right)$ se trouve dans la case située sur la $i^{ème}$ ligne et $\iota^{ème}$ colonne de Ξ, comme illustré sur la Table 5.1.

Pour apparier chaque hyper-sommet de $H_h^{(1)}$ avec un et un seul de $H_h^{(2)}$, qui ne soit pas encore apparié, nous avons choisi d'utiliser la méthode Hongroise sur Ξ. Nous notons $|V_{max}| = \max(|V^{(1)}|, |V^{(2)}|)$, le nombre maximal d'hyper-sommets entre $H_h^{(1)}$ et $H_h^{(2)}$. Une condition nécessaire à l'utilisation de cet algorithme est que Ξ soit une matrice carrée d'ordre $|V_{max}|$. Pour ce faire, si $|V^{(1)}| < |V^{(2)}|$ (respectivement $|V^{(2)}| < |V^{(1)}|$) nous ajoutons autant de lignes (respectivement, colonnes) que nécessaire pour obtenir une matrice carrée. Chaque case nouvellement ajoutée est associée à la moyenne des valeurs de Ξ pour pénaliser la différence du nombre d'hyper-sommets. La matrice carrée obtenue, d'ordre $|V_{max}|$ est appelée $\Xi^{\square}$. La Table 5.2 illustre le cas où $|V^{(1)}| = |V^{(2)}| + 2 = |V_{max}|$.

Toute ligne supplémentaire (respectivement, colonne) ajoutée est associée à un hyper-sommet, dit "fantôme", de $H_h^{(1)}$, noté $v_f^{(1)}$ (respectivement, $H_h^{(2)}$, noté $v_f^{(2)}$). Un hyper-sommet fantôme n'existe pas à proprement parler. Nous choisissons donc de lui affecter la valeur d'importance 0 dans $H_h^{(1)}$ (respectivement, $H_h^{(2)}$). Il sera tout de même apparié à un hyper-sommet de $H_h^{(2)}$ (respectivement, $H_h^{(1)}$).

La matrice étant maintenant carrée, il suffit de choisir un seul élément par ligne et par

colonne de façon à rendre la somme des éléments choisis minimale. Pour ce faire, nous devons préciser qu'un ensemble $\mathcal{M}$ d'éléments d'une matrice est dit "*indépendant*" si aucun élément de $\mathcal{M}$ n'est situé sur la même ligne ou colonne qu'un autre élément de $\mathcal{M}$. La matrice est réduite en soustrayant, à chaque ligne, son plus petit élément puis, à chaque colonne, son plus petit élément également. Les deux étapes suivantes sont alors nécessaires :

1. Trouver $\mathcal{M}$ un ensemble indépendant maximal de zéros (l. 2 de l'Algorithme 14) en utilisant la méthode de Munkres (1957). Si cet ensemble a $|V_{max}|$ éléments, il s'agit de la solution optimale, sinon aller à l'étape (2).
2. Marquer la ligne et la colonne de chaque élément de $\mathcal{M}$. Soit ω le plus petit élément non marqué (c'est-à-dire dont la ligne et la colonne ne sont pas marquées). Soustraire ω à chaque élément de $\Xi^{\square}$ non marqué et ajouter ω à chaque élément de $\Xi^{\square}$ dont la ligne et la colonne sont toutes deux marquées (l. 3 à 18 de l'Algorithme 14).

L'algorithme hongrois retourne une liste de sommets appariés $\mathcal{M} = \{(v_1^{(1)}, v_\omega^{(2)}), \ldots, (v_i^{(1)}, v_{\omega'}^{(2)}), \ldots, (v_{|V_{max}|}^{(1)}, v_{\omega''}^{(2)})\}$ où $1 < i < |V_{max}|$ et $\omega, \omega', \omega''$ appartiennent$[1, |V_{max}|]$ avec $\omega \neq \omega' \neq \omega''$.

L'intérêt de la méthode hongroise à ce niveau-là est qu'elle résout les problèmes d'affectation minimale. Dans notre cas, il s'agit d'appairer les hyper-sommets qui ont le moins de dissimilarité (ou le plus petit coût d'appariement). L'appariement des hyper-sommets est important à cette étape puisque chaque hyper-sommet représente une partie de la forme, cela revient à apparier les parties de forme.

5.1.3 Algorithme et exemple de l'étape d'initialisation

Pour résumer l'étape d'initialisation, nous proposons d'illustrer l'Algorithme 15, reprenant les étapes précédentes plus globalement, grâce à un exemple présenté sur la Figure 5.7. Les deux hyper-graphes à apparier sont issus des Figures 5.6(a) et (b). Les méta-sommets sont constitués de leurs méta-branches. Le coût d'appariement des hyper-sommets $v_1^{(1)}$ et $v_1^{(2)}$ est calculé grâce à la distance du cantonnier sur la matrice $\psi\left(v_1^{(1)}, v_1^{(2)}\right)$, celle ayant le fond turquoise. En effectuant cette opération sur chaque paire d'hyper-sommets, nous obtenons la matrice Ξ. Puis, nous la transformons en matrice carrée et la réduisons pour obtenir $\Xi^{\square}$. Enfin, nous appliquons l'algorithme hongrois pour obtenir l'appariement des hyper-sommets (mis en évidence dans $\Xi^{\square}$).

5.1.4 Raffinement de l'appariement entre hyper-graphes : procédure de mise à jour

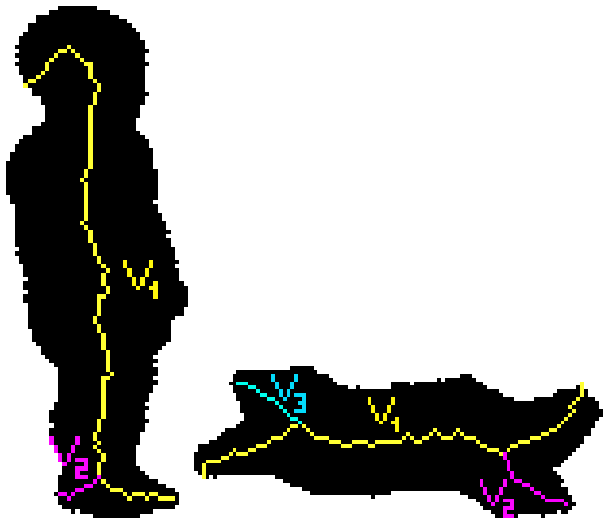
La procédure de mise à jour utilise l'appariement $\mathcal{M}$, donné par l'étape d'initialisation, pour l'affiner. Elle est notamment utilisée par Kim et al. (2010). L'idée générale est d'obtenir un appariement d'hyper-sommets terminal qui soit en corrélation avec l'appariement utilisé pour calculer le coût d'appariement entre deux hyper-sommets. L'idée est d'utiliser $\mathcal{M}$ sur ψ, en tenant compte de l'importance des branches grâce à une fonction de pénalisation (Figure 5.5),

Algorithme 14: Algorithme hongrois.

Entrées : $\Xi^{\square}$: matrice carrée réduite

Sorties : $\mathcal{M}$: ensemble de coordonnées dans $\Xi^{\square}$

début
  $\mathcal{M} \leftarrow$ *un ensemble indépendant de zéros de taille maximale dans* $\Xi^{\square}$
  tant que $|\mathcal{M}| < |V_{max}|$ **faire**
    pour *Chaque élément* (l, c) *de* $\mathcal{M}$ **faire**
      Marquer la ligne l *et la colonne* c
      $\omega \leftarrow$ la plus petite valeur non marquée dans $\Xi^{\square}$
      pour l' *allant de* 1 *à* $|V_{max}|$ **faire**
        pour c' *allant de* 1 *à* $|V_{max}|$ **faire**
          si $\Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right)$ *n'est pas marqué* **alors**
            $\Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right) = \Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right) - \omega$
          fin
          si $\Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right)$ *est marqué deux fois* **alors**
            $\Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right) = \Xi^{\square}\left(v_{l'}^{(1)}, v_{c'}^{(2)}\right) + \omega$
          fin
        fin
      fin
    fin
  fin
  retourner $\mathcal{M}$
fin



(a) $S_h^{(1)}$ (b) $S_h^{(2)}$

FIGURE 5.6 – *Squelettes hiérarchiques de deux formes utilisés pour illustrer l'Algorithme 15.*

Algorithme 15: Algorithme de la procédure d'initialisation de notre méthode d'appariement.

Entrées : $H_h^{(1)}$, $H_h^{(2)}$
Sorties : ψ, $\Xi^{\square}$, $\mathcal{M}$

1 **début**
2 *Générer $\boldsymbol{\psi}$ en utilisant la Définition 5.4*
3 */* Construction de Ξ grâce à l'EMD */*
4 **pour** *i allant de 1 à* $|V^{(1)}|$ **faire**
5 **pour** *ι allant de 1 à* $|V^{(2)}|$ **faire**
6 *Calculer* $\Xi\left(v_i^{(1)}, v_\iota^{(2)}\right)$ *grâce à l'EMD et à la fonction de pénalisation // Équation 5.10*
7 **fin**
8 **fin**
9 */* Appariement des hyper-sommets grâce à l'algorithme hongrois sur $\Xi^{\square}$ */*
10 $\Xi^{\square} \leftarrow matriceCarreeReduite(\Xi)$
11 $\mathcal{M} = algoHongrois(\Xi^{\square})$*// Algorithme 14*
12 **fin**

$\psi\left(v_1^{(1)}, v_1^{(2)}\right) =$ (l. 2 Algo 15)

$\psi(v_1^{(1)} \smile v_1^{(1)}, v_1^{(2)} \smile v_1^{(2)}) = 22,6$	$\psi(v_1^{(1)} \smile v_1^{(1)}, v_1^{(2)} \smile v_2^{(2)}) = 75,2$	$\psi(v_1^{(1)} \smile v_1^{(1)}, v_1^{(2)} \smile v_3^{(2)}) = 75,1$
$\psi(v_1^{(1)} \smile v_2^{(1)}, v_1^{(2)} \smile v_1^{(2)}) = 75,9$	$\psi(v_1^{(1)} \smile v_2^{(1)}, v_1^{(2)} \smile v_2^{(2)}) = 19,2$	$\psi(v_1^{(1)} \smile v_2^{(1)}, v_1^{(2)} \smile v_3^{(2)}) = 18,6$

$emd(\omega_1^{(1)}, \omega_1^{(2)}) \times$ *fonction de pénalisation (Équation 5.10)*

$\Xi =$ (l. 3 à 8 Algo 15)

$\Xi\left(v_1^{(1)}, v_1^{(2)}\right) = 20,6$	$\Xi\left(v_1^{(1)}, v_2^{(2)}\right) = 73,5$	$\Xi\left(v_1^{(1)}, v_3^{(2)}\right) = 73,1$
$\Xi\left(v_2^{(1)}, v_1^{(2)}\right) = 73,9$	$\Xi\left(v_2^{(1)}, v_2^{(2)}\right) = 17,5$	$\Xi\left(v_2^{(1)}, v_3^{(2)}\right) = 16,6$

Obtention d'une matrice carrée

$\Xi^{\square} =$ (l. 9-10 Algo 15)

$\Xi^{\square}\left(v_1^{(1)}, v_1^{(2)}\right) = 20,6$	$\Xi^{\square}\left(v_1^{(1)}, v_2^{(2)}\right) = 73,5$	$\Xi^{\square}\left(v_1^{(1)}, v_3^{(2)}\right) = 73,1$
$\Xi^{\square}\left(v_2^{(1)}, v_1^{(2)}\right) = 73,9$	$\Xi^{\square}\left(v_2^{(1)}, v_2^{(2)}\right) = 17,5$	$\Xi^{\square}\left(v_2^{(1)}, v_3^{(2)}\right) = 16,6$
$\Xi^{\square}\left(v_f^{(1)}, v_1^{(2)}\right) = 45,9$	$\Xi^{\square}\left(v_f^{(1)}, v_2^{(2)}\right) = 45,9$	$\Xi^{\square}\left(v_f^{(1)}, v_3^{(2)}\right) = 45,9$

$\mathcal{M} = \{(v_1^{(1)}, v_1^{(2)}), (v_2^{(1)}, v_3^{(2)}), (v_f^{(1)}, v_2^{(2)})\}$

FIGURE 5.7 – *Illustration de l'Algorithme 15 à partir de l'exemple des deux formes de la Figure 5.6.*

pour déterminer la dissimilarité affinée. En d'autres termes, la correspondance précédemment déterminée $\mathcal{M}$ est utilisée au lieu d'exécuter la distance du cantonnier comme précédemment. Plus précisément, un nouveau coût d'appariement entre chaque paire d'hyper-sommets $v_i^{(1)}$ et $v_\iota^{(2)}$ est calculé grâce à l'équation 5.11, noté $\overline{\Xi}\left(v_i^{(1)}, v_\iota^{(2)}\right)$. La matrice formée est alors appelée $\overline{\Xi}$ tel que :

$$\overline{\Xi} = \left[\overline{\Xi}\left(v_i^{(1)}, v_\iota^{(2)}\right)\right]_{\substack{0\leq i\leq |V^{(1)}|\\ 0\leq \iota\leq |V^{(2)}|}}$$

L'algorithme hongrois est alors exécuté sur cette matrice. Si le nouvel appariement d'hyper-sommets obtenu est le même que précédemment, nous arrivons à une stabilité. Dans le cas contraire, cette procédure est répétée jusqu'à atteindre une stabilité.

$$\overline{\Xi}\left(v_i^{(1)}, v_\iota^{(2)}\right) = \sum_{(v_{i'}^{(1)}, v_{\iota'}^{(2)})\in\mathcal{M}} \psi\left(v_i^{(1)} \smile v_{i'}^{(1)}, v_\iota^{(2)} \smile v_{\iota'}^{(2)}\right)\left(e^{2|w(v_i^{(1)})-w(v_\iota^{(2)})|} + (2 - w(v_i^{(1)}) - w(v_\iota^{(2)}))\right) \tag{5.11}$$

De cette manière, nous obtenons une cohérence entre les mesures utilisées (géométrie, contexte) pour calculer les valeurs de dissimilarité entre chaque sommet et l'appariement final. Le pseudo-code est détaillé dans l'Algorithme 16.

Algorithme 16: Algorithme de mise à jour de notre méthode d'appariement.

Entrées : $\boldsymbol{\psi}$, $\mathcal{M}$
Sorties : $\overline{\Xi}$, $\overline{\mathcal{M}}$

```
début
    M̄ ← M ; M̃ ← {} ;
    tant que M̄ ≠ M̃ faire
        M̃ ← M̄
        Calculer Ξ̄ grâce à M̃ sur ψ // Équation 5.11
        Ξ̄^□ ← matriceCarreeReduite(Ξ̄)
        M̄ ← algoHongrois(Ξ̄^□) // Algorithme 14
    fin
fin
```

5.1.5 Mesure de dissimilarité globale entre deux hyper-graphes

La mesure de dissimilarité globale entre deux hyper-graphes se calcule à partir de $\overline{\Xi}$, de $\overline{\mathcal{M}}$ et des valeurs d'importance associées à chaque hyper-sommet. En d'autres termes, nous prenons en compte la dissimilarité trouvée entre les hyper-sommets qui ont été appariés ainsi que leur importance.

Définition 5.5 *La dissimilarité globale entre deux hyper-graphes* $\mathrm{Y}_{H_h^{(2)}}^{H_h^{(1)}}$ *est définie par :*

$$\mathrm{Y}\left(H_h^{(1)}, H_h^{(2)}\right) = \frac{\sum_{(v_i^{(1)}, v_l^{(2)}) \in \mathcal{M}} \overline{\Xi}^{\square}\left(v_i^{(1)}, v_l^{(2)}\right) \max(w(v_i^{(1)}), w(v_l^{(2)}))}{\sum_{(v_i^{(1)}, v_l^{(2)}) \in \mathcal{M}} \max(w(v_i^{(1)}), w(v_l^{(2)}))}$$

Dans la partie suivante, nous verrons comment utiliser cette mesure de dissimilarité globale entre deux hyper-graphes pour connaître la forme provenant d'une base de données la plus proche d'une *forme requête*. Ensuite, nous présenterons la méthode à laquelle nous comparons notre travail puis une analyse de la complexité et des paramètres de la méthode proposée ainsi que les résultats obtenus sur différentes bases de données.

5.2 Résultats

Soit $\mathcal{B}$ une base de données dans laquelle $|\mathcal{B}|$ représente le nombre de formes y figurant.

$$\mathcal{B} = \{\mathcal{F}^{(\mathcal{B}_1)}, \ldots, \mathcal{F}^{(\mathcal{B}_{|\mathcal{B}|})}\}$$

et $\mathcal{F}^{(r)}$, la *forme requête*.

Pour déterminer les η formes les plus proches de la *forme requête*, nous supposons que les hyper-graphes hiérarchiques des formes appartenant à $\mathcal{B}$ ont été calculés préalablement en utilisant DECS comme squelette de base. La première étape effective est donc de calculer l'hyper-graphe hiérarchique de $\mathcal{F}^{(r)}$, à partir de DECS, une seule fois avant l'appariement à l'ensemble des formes de la base de données. Puis, il est nécessaire de calculer la valeur de dissimilarité entre $H_h^{(r)}$ et chaque hyper-graphe hiérarchique associé à chaque forme de $\mathcal{B}$. Les η formes les plus proches de $\mathcal{F}^{(r)}$ sont les formes associées aux η valeurs de plus faible dissimilarité. En d'autres termes, nous classons les formes par dissimilarité croissante.

5.2.1 Méthodes utilisées lors de la comparaion

Méthode d'appariement de graphes basés squelettes utilisant le chemin de similarité (Bai et Latecki (2008), Shen et al. (2013))

Les auteurs proposent un algorithme d'appariement de squelettes basé sur la comparaison des chemins géodésiques entre les points terminaux de chaque squelette. Cela leur permet de gérer les topologies plus complexes que celles des formes simplement connexes. En effet, si la forme contient des trous, il existe plusieurs chemins entre deux points terminaux donnés.

Leur but est donc d'apparier deux squelettes en établissant une correspondance entre les nœuds terminaux uniquement, qui encodent les zones saillantes du contour (*cf.* Figure 5.8). Autrement dit, les branches terminales peuvent être vues comme les parties visuelles de la forme d'origine.

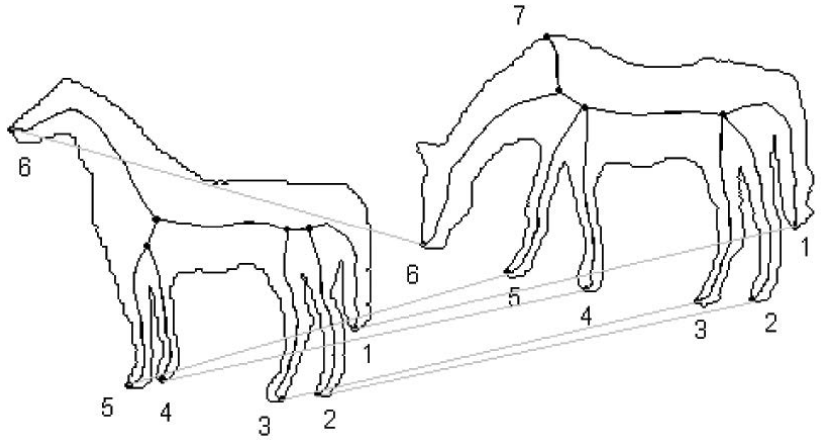


FIGURE 5.8 – *Appariement des nœuds terminaux de deux squelettes selon Bai et Latecki (2008).*

Pour apparier les nœuds terminaux de deux squelettes $S^{(1)}$ et $S^{(2)}$, ils commencent par les ordonner dans le sens anti-trigométrique. Puis, pour chaque paire de nœuds terminaux $(\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)})$, tel que $\boldsymbol{p}^{(1)}$ (respectivement, $\boldsymbol{p}^{(2)}$) appartient à $S^{(1)}$ (respectivement, $S^{(2)}$), ils calculent la distance de chemin entre tous les chemins géodésiques reliant $\boldsymbol{p}^{(1)}$ (respectivement, $\boldsymbol{p}^{(2)}$) à tous les autres nœuds terminaux de $S^{(1)}$ (respectivement, $S^{(2)}$). Les résultats sont alors regroupés dans une matrice. Le chemin géodésique entre deux points terminaux n'est autre que le plus court chemin entre ces deux points en n'utilisant que les points du squelette. Pour trouver une distance entre deux chemins géodésiques (distance de chemin), les auteurs commencent par les échantillonner tous les deux avec k points équidistants et utilisent la somme des différences de rayon normalisé entre chaque paire de points correspondants, qu'ils ajoutent à une différence de longueur pondérée.

Pour chaque matrice issue de $(\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)})$, ils appliquent l'algorithme *OSB* (Optimal Subsequence Bijection) de Latecki et al. (2007b) afin d'obtenir une valeur de dissimilarité entre $\boldsymbol{p}^{(1)}$ et $\boldsymbol{p}^{(2)}$. La méthode *OSB* a pour but d'apparier les chemins géodésiques issus de $\boldsymbol{p}^{(1)}$ et $\boldsymbol{p}^{(2)}$ en respectant le sens horaire mais en s'autorisant à sauter des chemins si leur appariement est trop aberrant.

À l'issue de cette étape, les auteurs obtiennent une valeur de dissimilarité pour chaque paire de nœuds terminaux qu'ils regroupent dans une matrice. Ils appliquent alors l'algorithme hongrois sur cette matrice pour associer les nœuds terminaux deux à deux.

Cet algorithme, publié dans un journal de référence (*Pattern Recognition*) peu avant le début de ma thèse, obtient d'excellents taux de réussite.

Au niveau de la complexité de l'algorithme, il a été prouvé par les auteurs qu'elle était en $O(|N_t^{(1)}|^2 \times |N_t^{(2)}|^2)$, où $N_t^{(1)}$ et $N_t^{(2)}$ sont les ensembles de nœuds terminaux et de jonction dans $S^{(1)}$ et $S^{(2)}$.

Méthode utilisant le contexte de formes (Belongie et al. (2002))

Belongie et al. (2002) ont proposé les contextes de formes (*shape context*). Ils échantillonnent le bord de la forme et décrivent la distribution spatiale relative (distance euclidienne et orien-

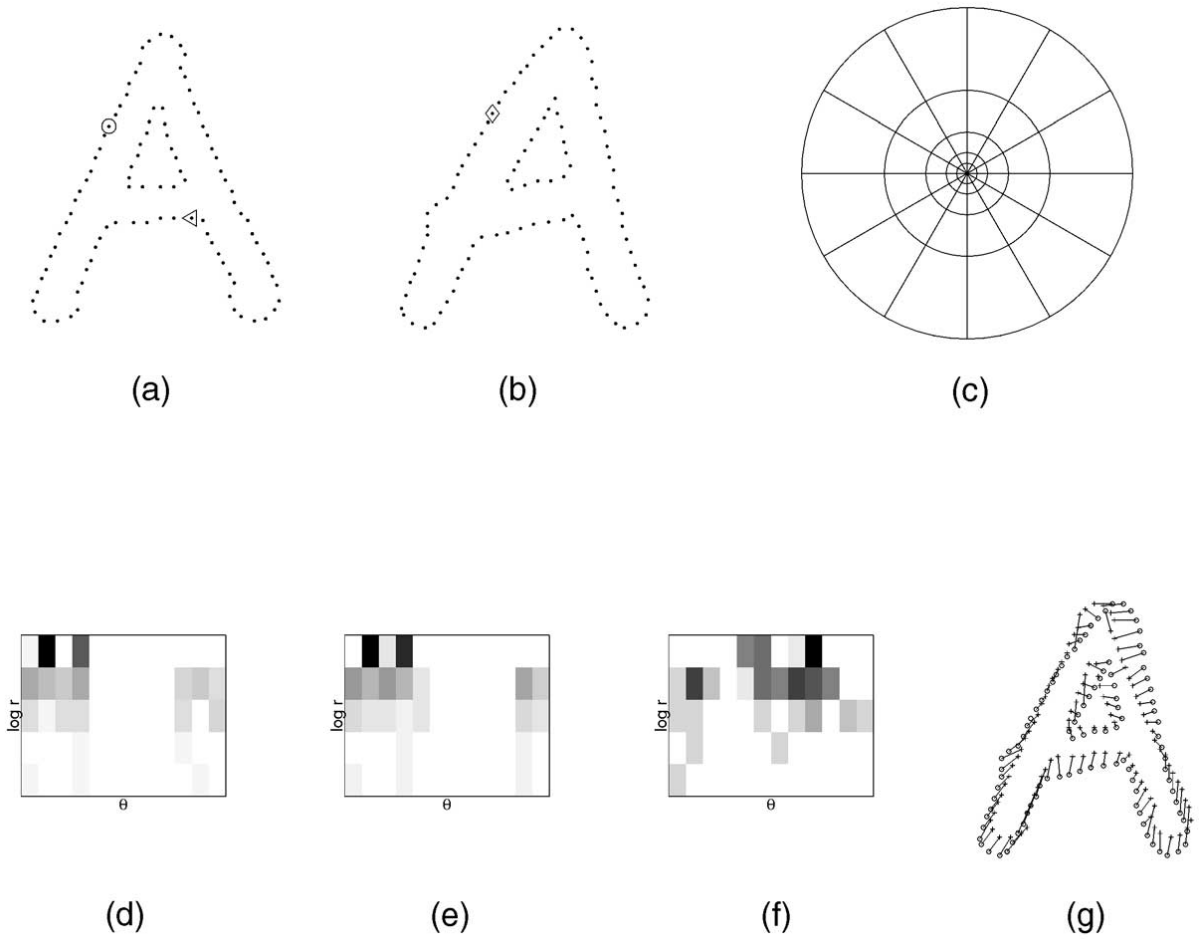


FIGURE 5.9 – *Contexte de forme (Belongie et al. (2002)). (a) et (b) Échantillonnage du bord de deux formes. (c) Diagramme d'histogrammes log-polaire. (d), (e) et (f) Exemples de contextes de forme marqués par les symboles* $\circ$, $\diamond$ *et* $\triangleleft$ *sur les figures (a) et (b). (g) Appariement des formes (a) et (b).*

tation) des points. Pour ce faire, ils construisent un diagramme relativement grossier (pour chacun des points échantillonnés) appelé le contexte de forme en positionnant le centre du diagramme d'histogrammes log-polaire sur un des points et en y reportant le nombre de points échantillonnés obtenus dans chaque case. L'ensemble des contextes de formes pour chacun d'eux est ensuite utilisé pour apparier les formes. La figure 5.9 illustre cette méthode. Cette méthode a été choisie car elle n'utilise pas le squelette comme descripteur de formes et elle a également été utilisée par Bai et Latecki (2008) pour évaluer leur travail. Par ailleurs, cela nous permet de positionner notre travail de façon plus globale dans la littérature. Enfin, elle a été publiée dans un très bon journal (*Pattern Analysis and Machine Intelligence*).

5.2.2 Complexité

Dissimilarité entre deux hyper-sommets *via* la distance du cantonnier

La distance de branches entre chaque paire d'hyper-sommets provenant de deux hypergraphes peut être calculée une seule fois pour chaque paire. La complexité est alors en

$O(|V^{(1)}||V^{(2)}|)$ puisque chaque hyper-sommet de $V^{(1)}$ est comparé à chaque hyper-sommet de $V^{(2)}$.

Le calcul du plus court chemin entre chaque paire d'hyper-arêtes d'un même graphe peut également être réalisé une seule fois et stocké pour être réutilisé. En théorie, l'algorithme de Dijkstra a une complexité algorithmique en $O((|E|+|V|)ln|E|)$ pour calculer le plus court chemin entre une hyper-arête et chaque hyper-arête du même hyper-graphe. Ainsi, pour calculer le plus court chemin entre chaque paire d'hyper-arêtes, il faut lancer l'algorithme de Dijkstra $|E|$ fois, où $|E|$ est le nombre d'hyper-arêtes dans l'hyper-graphe H.

La complexité algorithmique de la distance du cantonnier (EMD) est en $O(|V^{(1)}||V^{(2)}|CS(|V^{(1)}|,|V^{(2)}|))$ où $O(|V^{(1)}||V^{(2)}|$ est la complexité du calcul de $\left[\psi^{v_i^{(1)} \smile v_{i'}^{(1)}}_{v_t^{(2)} \smile v_t^{(2)}}\right]_{\substack{0\leq i'\leq |V^{(1)}| \\ 0\leq \iota'\leq |V^{(2)}|}}$ à partir des calculs de distance de branche et de plus court chemin stockés en amont et, où CS est la complexité de l'algorithme du simplexe en fonction de $|V^{(1)}|$ et $|V^{(2)}|$. Alors que le simplexe est un algorithme théoriquement exponentiel, il a été montré dans Spielman et Teng (2004) qu'il est "en pratique" en temps polynomial : $CS(|V^{(1)}|,|V^{(2)}|) = O(\max(|V^{(1)}|,|V^{(2)}|))$. Ainsi, le calcul de la dissimilarité entre deux hyper-sommets *via* la distance du cantonnier est en $O(|V^{(1)}||V^{(2)}|) + O(\max(|V^{(1)}|^2 ln|V^{(1)}|,|V^{(2)}|^2 ln|V^{(2)}|)) + O(|V^{(1)}||V^{(2)}|\max(|V^{(1)}|,|V^{(2)}|)$, soit en $O(|V^{(1)}||V^{(2)}|\max(|V^{(1)}|,|V^{(2)}|)$ puisque la complexité du calcul des distances de branche et des plus courts chemins est négligeable par rapport à la complexité de l'EMD.

Par conséquent, la complexité entre chaque paire d'hyper-sommets est en $O(|V^{(1)}|^2|V^{(2)}|^2\max(|V^{(1)}|,|V^{(2)}|)$

Appariement initial *via l'algorithme hongrois*

La complexité de l'algorithme Hongrois original est en $O(\max(|V^{(1)}|,|V^{(2)}|)^4)$ (Kuhn (1955)). Néanmoins, la complexité a été réduite à $O(\max(|V^{(1)}|,|V^{(2)}|)^3)$ grâce à diverses méthodes comme celle de Lawler (2001), par exemple. Elle est donc négligeable comparée à la complexité vue dans le paragraphe précédent.

Procédure de mise à jour

Il a été déterminé expérimentalement, par Kim et al. (2010), que la complexité de cette procédure de mise à jour était fortement négligeable par rapport aux étapes précédentes.

Complexité totale

Comme nous l'avons vu dans les deux chapitres précédents, le calcul du squelette est linéaire en $O(n)$ et sa hiérarchisation est en $O(n^2)$. En terme de nombre de branches, la complexité de l'appariement d'une forme à une autre forme provenant d'une base de données est $O(|V^{(1)}|^2|V^{(2)}|^2\max(|V^{(1)}|,|V^{(2)}|)$.

FIGURE 5.10 – *Base de données Kimia 216 (Sebastian et al. 2004).*

5.2.3 Résultats expérimentaux

Nous avons réalisé nos principaux tests sur la base de données Kimia216 (Sebastian et al. (2004)). Il s'agit d'une base de données composée de 18 catégories ayant chacune 12 formes. Cette base est présentée sur la Figure 5.10. Nous considérons chaque forme de cette base comme une *forme requête*. Pour chacune d'elles, le but est de retrouver les onze autres formes appartenant à la même catégorie. Les squelettes utilisés ont été générés grâce à la méthode DECS en utilisant les seuils déterminés expérimentalement (*cf. Chapitre 3*) qui permettent d'obtenir une bonne résistance au bruit, à savoir $th_{perc-max-ball} = 0.4$ et $th_{ridge-high} = 0.5$.

Variation des paramètres

La méthode d'appariement proposée nécessite trois paramètres α, β et h. Afin de les optimiser et de montrer leur influence sur la reconnaissance, nous avons procédé à une étude de

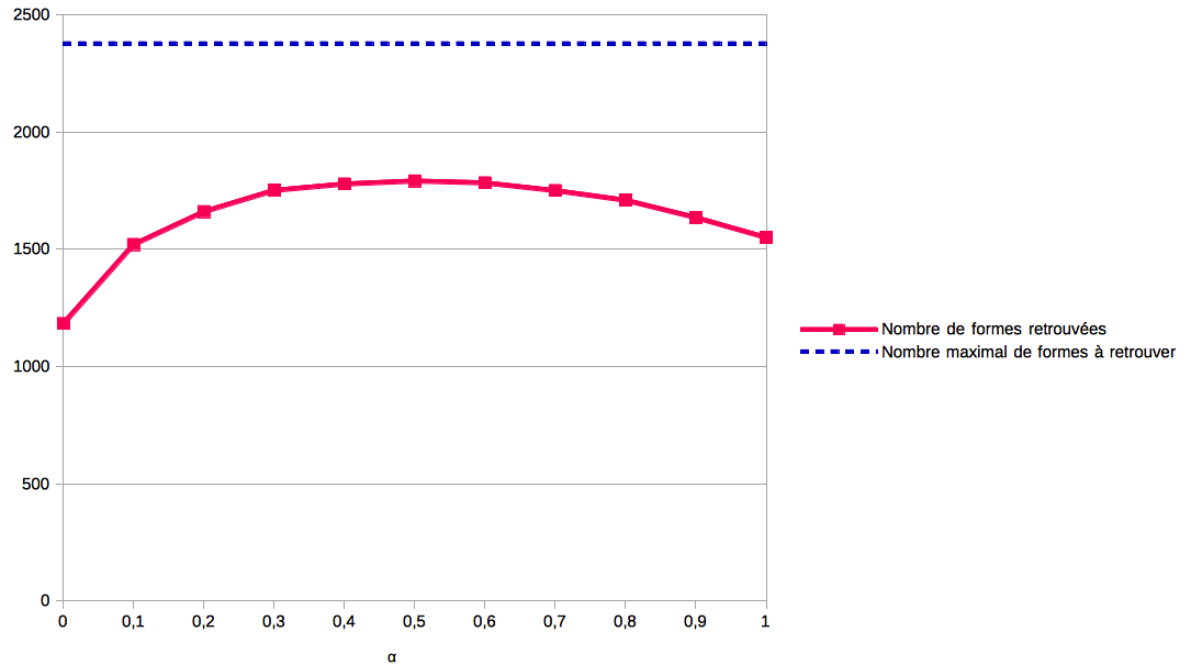


FIGURE 5.11 – *Nombre de formes retrouvées en fonction du poids α.*

leur impact sur les résultats.

Détermination du poids α :
Pour commencer, nous avons voulu étudier l'influence du poids α qui permet de déterminer la proportion entre les deux termes présents dans le calcul de la dissimilarité géométrique (la différence de longueur et la différence d'épaisseur le long des méta-branches). Pour ce faire, nous avons annulé le terme correspondant au contexte, c'est-à-dire la position relative d'une méta-branche par rapport aux autres méta-branches du squelette hiérarchique. Cela revient à prendre $\beta = 1$ dans la définition 5.4. Nous avons également échantillonné chaque méta-branche avec $k = 50$ points.

Les résultats de la variation du poids α sont présentés sur la Figure 5.11. Il s'agit d'un graphique représentant le nombre total de formes retrouvées en fonction de α. Étant donné qu'il y a 216 formes requêtes et que, pour chacune d'elles il est possible de reconnaître au maximum onze formes de la même catégorie, le nombre maximal de formes retrouvées vaut $216 * 11 = 2376$ formes. Nous observons sur la figure que les meilleurs résultats sont obtenus avec $\alpha = 0.5$. Nous avons donc fixé ce poids à 0.5 par la suite. Cela revient à dire que la différence de longueur entre deux branches a autant d'importance que la différence d'épaisseur le long de ces dernières.

Détermination du paramètre β :
Nous déterminons maintenant dans quelle proportion tenir compte du contexte. α ayant été fixé précédemment, nous conservons $k = 50$ et faisons varier β. La Figure 5.12 présente les

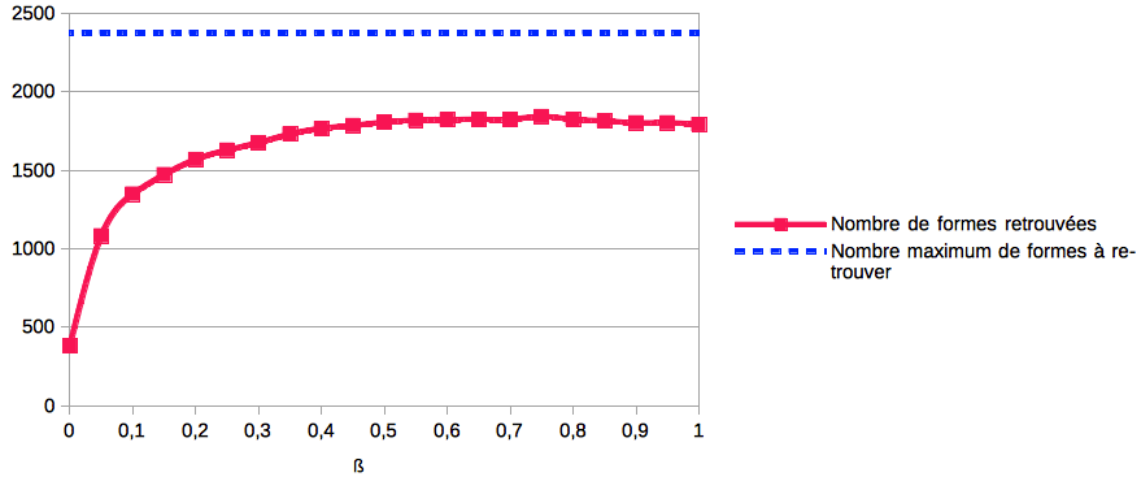


FIGURE 5.12 – *Nombre de formes retrouvées en fonction du poids β.*

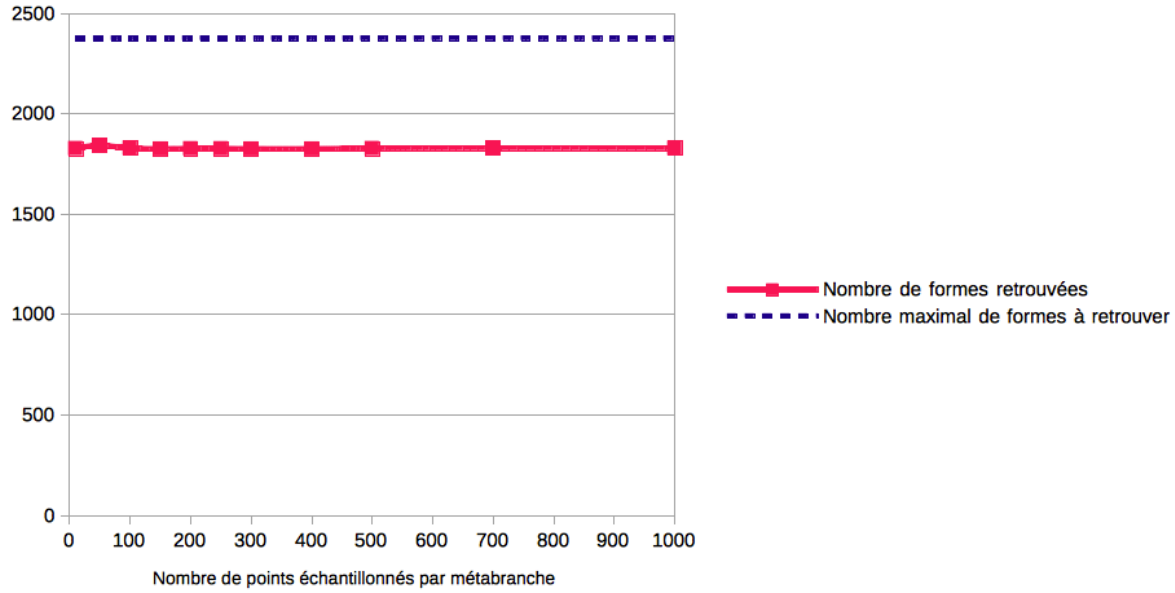


FIGURE 5.13 – *Nombre de formes retrouvées en fonction du paramètre k.*

résultats obtenus concernant le nombre de formes retrouvées en fonction de β. Les meilleurs résultats sont obtenus pour β égal à 0.75, et ne se dégradent pas significativement lorsque ce poids augmente. Ceci signifie que la géométrie de la forme est prépondérante par rapport au contexte. Nous avons donc choisi cette valeur pour la suite.

Détermination du paramètre k, le nombre de points échantillonnés par méta-branche :
Le dernier paramètre que nous pouvons tester est k le nombre de points échantillonnés par branche. Pour ce faire, nous conservons les poids α et β déterminés précédemment et faisons évoluer k. La Figure 5.13 représente la courbe du nombre de formes retrouvées en fonction de k. Nous pouvons remarquer que l'influence du nombre de points échantillonnés par branche est minime au regard des autres paramètres. Nous fixons donc k à 50 pour la suite car, il est inutile de prendre plus de points, cela alourdirait inutilement le traitement.

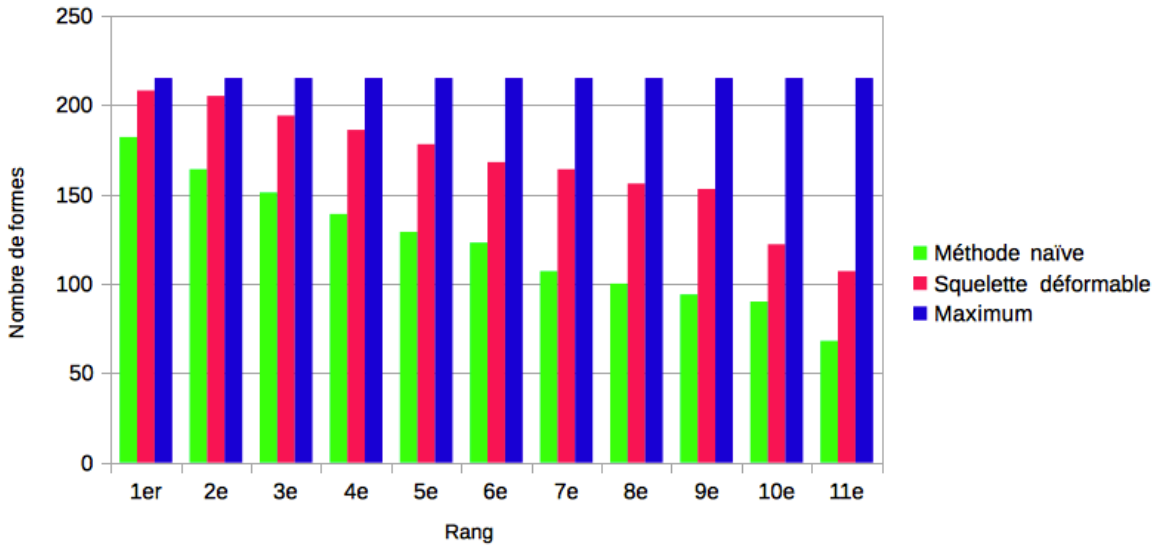


FIGURE 5.14 – *Nombre de formes retrouvées, à chaque rang, en utilisant la méthode naïve et le squelette déformable.*

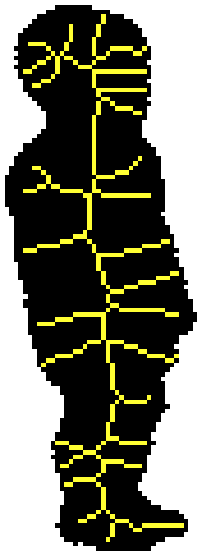
En résumé, dans la suite de nos travaux, nous utiliserons $\alpha = 0.5$, $\beta = 0.75$ et $k = 50$.

Méthode naïve *vs.* squelette déformable

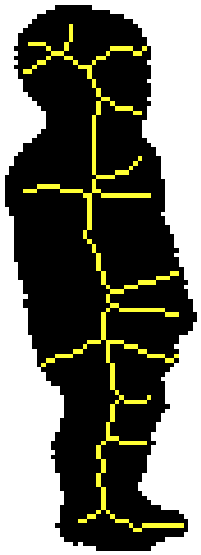
Afin de montrer la supériorité, en matière d'appariement, de la méthode basée sur le squelette déformable par rapport à la méthode naïve, nous avons procédé à l'appariement des formes de la base Kimia 216 en utilisant chacune des deux méthodes avec des paramètres identiques, fixés précédemment. La Figure 5.14 fournit, pour chaque rang, le nombre de formes correctement appariées. Plus précisément, pour une *forme requête*, nous cherchons les onze formes les plus similaires de la base de données. Puis, pour i compris entre 1 et 11, nous regardons si la i_e forme trouvée appartient à la même catégorie que la *forme requête*. Si c'est le cas, nous considérons que la forme est bien classée au i_e rang, dans le cas contraire, elle est considérée comme mal classée. On note donc en observant la figure que la méthode naïve obtient de moins bons résultats que la méthode basée sur le squelette déformable.

Utilisation de squelettes bruités

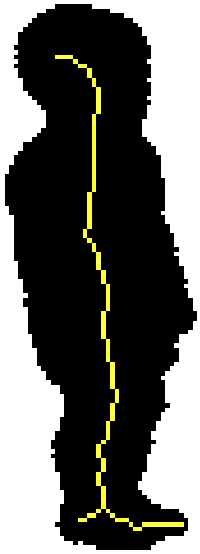
Dans cette partie, nous avons voulu tester notre méthode d'appariement sur des squelettes très bruités, c'est-à-dire que nous avons diminué les seuils de DECS de façon à réduire considérablement le nombre de branches élaguées. Pour ce faire, nous avons réalisé l'appariement en conservant les mêmes paramètres d'appariement que précédemment. Seuls ceux concernant le squelette ont été modifiés. Le squelette 1 a pour seuils $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.1$, le squelette 2 $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.2$ et le squelette 3 $th_{perc-max-ball} = 0.4$ et $th_{ridge-high} = 0.5$. Notons que le squelette 1 n'est presque pas élagué avec les seuils que nous lui avons fixés. Un exemple de squelettes obtenus avec ces seuils est présenté sur la Figure 5.15. Les résultats de l'appariement sont présentés sur la Figure 5.16.



(a) $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.1$



(b) $th_{perc-max-ball} = 0.1$ et $th_{ridge-high} = 0.2$



(c) $th_{perc-max-ball} = 0.4$ et $th_{ridge-high} = 0.5$

FIGURE 5.15 – *Exemple de squelettes à différents seuils.*

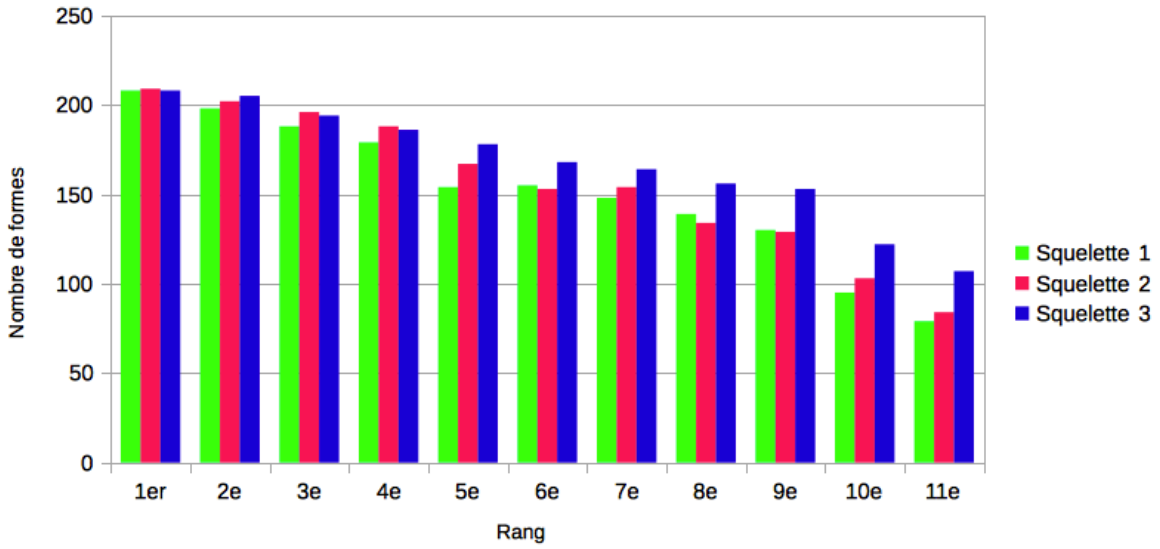


FIGURE 5.16 – *Nombre de formes retrouvées, à chaque rang, en utilisant trois couples de seuils dans la construction des squelettes grâce à la méthode DECS. Le squelette 1 a pour seuils* $th_{perc-max-ball} = 0.1$ *et* $th_{ridge-high} = 0.1$, *le squelette 2* $th_{perc-max-ball} = 0.1$ *et* $th_{ridge-high} = 0.2$ *et le squelette 3* $th_{perc-max-ball} = 0.4$ *et* $th_{ridge-high} = 0.5$.

Nous observons que les résultats sont moins probants mais restent tout à fait convenables. En effet, il est normal que le bruit dégrade la qualité de l'appariement puisque celle du squelette est affectée. Cependant, le fait de donner des poids très faibles à ces branches permet à l'algorithme d'appariement d'obtenir des résultats satisfaisants. Cette expérience montre, par conséquent, l'intérêt du poids affecté à chaque branche de squelette, autrement dit, l'intérêt du squelette hiérarchique.

Comparaison avec d'autres méthodes et discussions

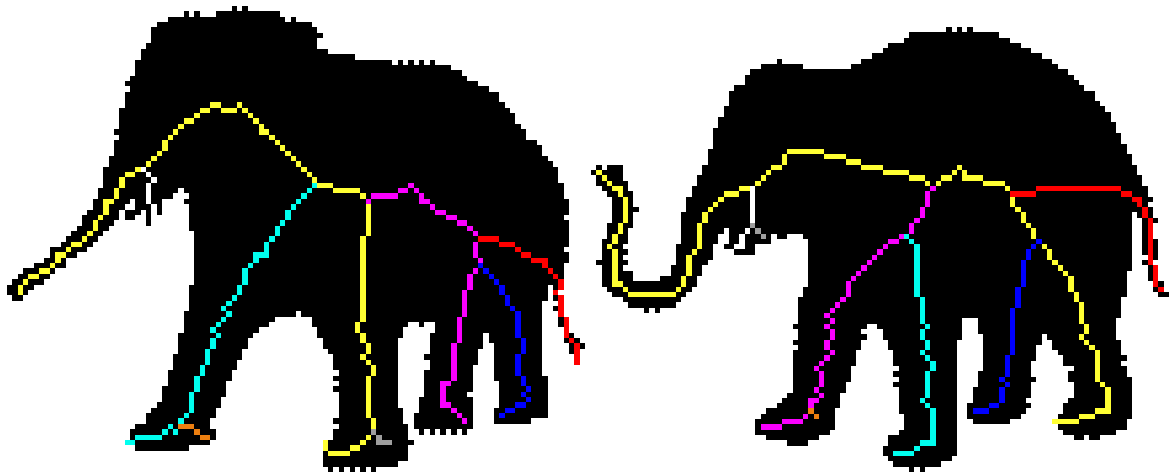
Afin de positionner notre travail par rapport à l'état de l'art, nous avons comparé notre méthode à deux autres de la littérature utilisant deux descripteurs de formes différents. La méthode d'appariement de graphes basés squelettes utilisant le chemin de similarité de Bai et Latecki (2008) est une méthode s'appuyant sur le squelette, proche de la méthode que nous proposons et publiée dans un journal de premier plan (*Pattern Recognition*). En revanche, la seconde méthode, celle de Belongie et al. (2002), est basée sur le contexte de formes, un tout autre descripteur de formes. Ces deux méthodes ont été décrites dans la Section 5.2.1.

Au vu des résultats figurant dans la Table 5.3, nous pouvons considérer que notre méthode fournit des résultats corrects et relativement proches de la méthode basée sur le contexte de forme. En revanche, ils sont un peu en-dessous de ceux de Bai et Latecki (2008) sur cette base de données. Nous avons plusieurs explications à cela. Rappelons que notre méthode est initialement inspirée de celle de Bai et Latecki (2008). Afin d'avoir une complexité inférieure à celle de leur méthode, nous avons voulu réduire le nombre de comparaisons entre branches.

En effet, pour chaque sommet terminal donné, Bai et Latecki (2008) créent autant de chemins qu'il y a de nœuds terminaux - 1, puis ils comparent les chemins deux à deux. Appelons

	1^{er}	2^{e}	3^{e}	4^{e}	5^{e}	6^{e}	7^{e}	8^{e}	9^{e}	10^{e}	11^{e}
Bai et Latecki (2008)	216	216	215	216	213	210	210	207	205	191	177
Belongie et al. (2002)	214	209	205	197	191	178	161	144	131	101	78
Méthode proposée	208	205	194	186	178	168	164	156	153	122	107

TABLE 5.3 – *Nombre de formes associées à une forme de la même catégorie de la base de données Kimia216 (Sebastian et al. (2004)) à chaque rang. Les 216 formes requêtes sont chacune des formes de la base de données Kimia216 (Sebastian et al. (2004))*



(a) (b)

FIGURE 5.17 – *Méta-branches obtenues sur deux formes appartenant à la même catégorie mais dont l'agencement des branches internes est différent*

$|N_t^{(1)}|$ (respectivement, $|N_t^{(2)}|$), le nombre de nœuds terminaux dans le squelette de la première forme (respectivement, de la deuxième forme). Dans leur cas, la complexité de cette étape est en $O(|N_t^{(1)}|^2|N_t^{(2)}|^2)$. La hiérarchisation de notre squelette permet d'éviter de comparer plusieurs fois une même branche de squelette comme ils le font. Ainsi, si nous considérons que $|V^{(1)}|$ (respectivement $|V^{(2)}|$), le nombre de méta-branches dans le squelette hiérarchique de la première (respectivement, deuxième) forme est du même ordre de grandeur que $|N_t^{(1)}|$ (respectivement, $|N_t^{(2)}|$), la complexité de la méthode proposée correspondant à cette étape est en $O(|N_t^{(1)}||N_t^{(2)}|)$. De plus, le calcul de la dissimilarité géométrique (db) est moins coûteux dans notre méthode puisque nos méta-branches sont plus courtes que leurs chemins. De ce point de vue, notre méthode est plus intéressante. Néanmoins, cela entraîne une moindre robustesse au réagencement des branches internes, comme le montre la Figure 5.17. Si la géométrie interne de la forme est différente, les méta-branches construites dans notre méthode seront également différentes. Le fait de tester tous les chemins, comme le font Bai et Latecki (2008), permet de ne pas obtenir ces erreurs.

Quant aux méthodes utilisées pour apparier les chemins (OSB) ou les méta-branches (EMD), il est très difficile de comparer leur complexité car les données traitées sont différentes. La complexité d'appariement des chemins est fonction du nombre de sommets alors que l'appariement des méta-branches est fonction du nombre de méta-branches. Cependant, la mé-

thode que nous utilisons semble moins efficace qu'OSB en terme de complexité. Néanmoins, pour notre structure basée sur les méta-branches, elle convient mieux puisqu'OSB n'est pas applicable du fait que nos méta-branches ne peuvent pas être organisées dans le sens horaire.

5.2.4 Application sur une base de données de feuilles d'arbres

Nous avons utilisé une base de feuilles construite dans le cadre de l'ANR ReVeS, elle-même générée à partir des bases réalisées par les organisateurs du challenge "Plant Identification" dans le cadre des conférences ImageClef 2011 et 2012 (Goëau et al. 2011; 2012). Notre méthode d'appariement étant normalisée, nous pouvons utiliser les paramètres que nous avons déterminés précédemment sur n'importe quelle base de données comportant des images de tailles variables. En moyenne, les images de la base Kimia 216 mesurent une centaine de pixels de hauteur alors celles de notre base de feuilles ont une hauteur de 800 pixels.

Dans cette base de données d'une vingtaine d'espèces et d'environ 170 images, certaines espèces n'étant représentées que par deux feuilles, nous avons cherché à déterminer l'appariement uniquement au premier rang. En d'autres termes, est-ce-que la feuille la plus similaire de la base est une feuille de la même espèce que la *feuille requête* ou pas. À noter que la difficulté majeure dans l'appariement de feuilles d'arbres est, que dans une même espèce, les feuilles peuvent avoir des formes complètement différentes (*cf.* Figure 5.19). *A contrario*, deux feuilles peuvent avoir une forme très ressemblante et pourtant ne pas appartenir à la même espèce (*cf.* Figure 5.20).

La Figure 5.18 montre les résultats obtenus. Grâce à la combinaison des méthodes de squelettisation, hiérarchisation et appariement que nous avons proposées, nous obtenons un taux de reconnaissance de 72%. Dans sa thèse, Cerutti (2013) utilise des descripteurs spécifiques aux feuilles et obtient un taux de reconnaissance de 42% en premier résultat et de 85% pour les cinq premiers. Ces résultats sont, néanmoins, à pondérer par le fait que sa base de données contient plus de classes (86 classes) et plus d'images que celle que nous avons utilisée. Les résultats obtenus peuvent donc être considérés comme bons. De plus, cela montre que la méthode proposée est utilisable dans un contexte réel. L'idée serait donc de combiner notre descripteur avec ceux qu'utilise Cerutti (2013), comme la forme de l'apex ou la forme de la base, pour tenter d'améliorer le taux de reconnaissance.

Les Figures proposées dans la Section 5.2.5 montrent quelques exemples d'appariements de feuilles associés aux valeurs de dissimilarités obtenues.

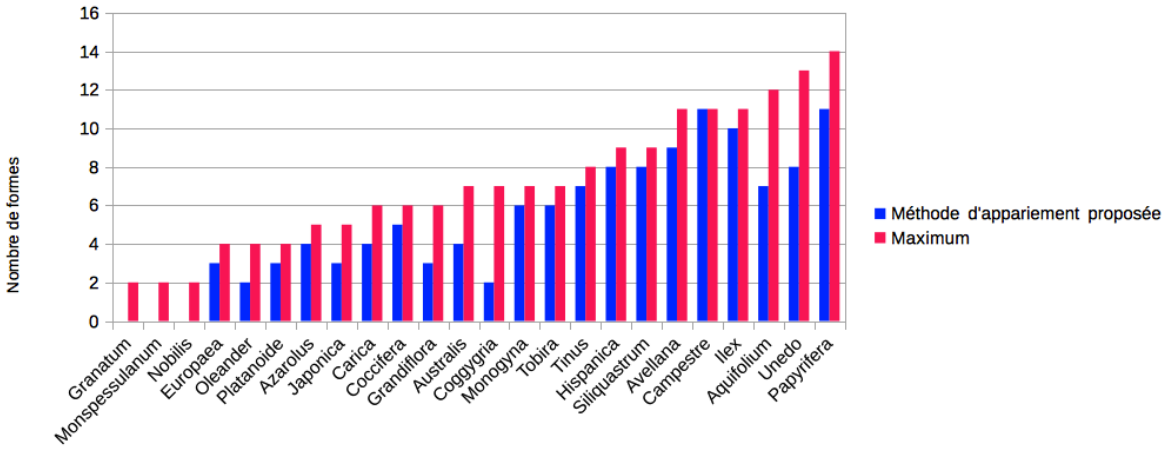


FIGURE 5.18 – *Nombre de formes retrouvées, au premier rang, pour chaque espèce de feuilles.*

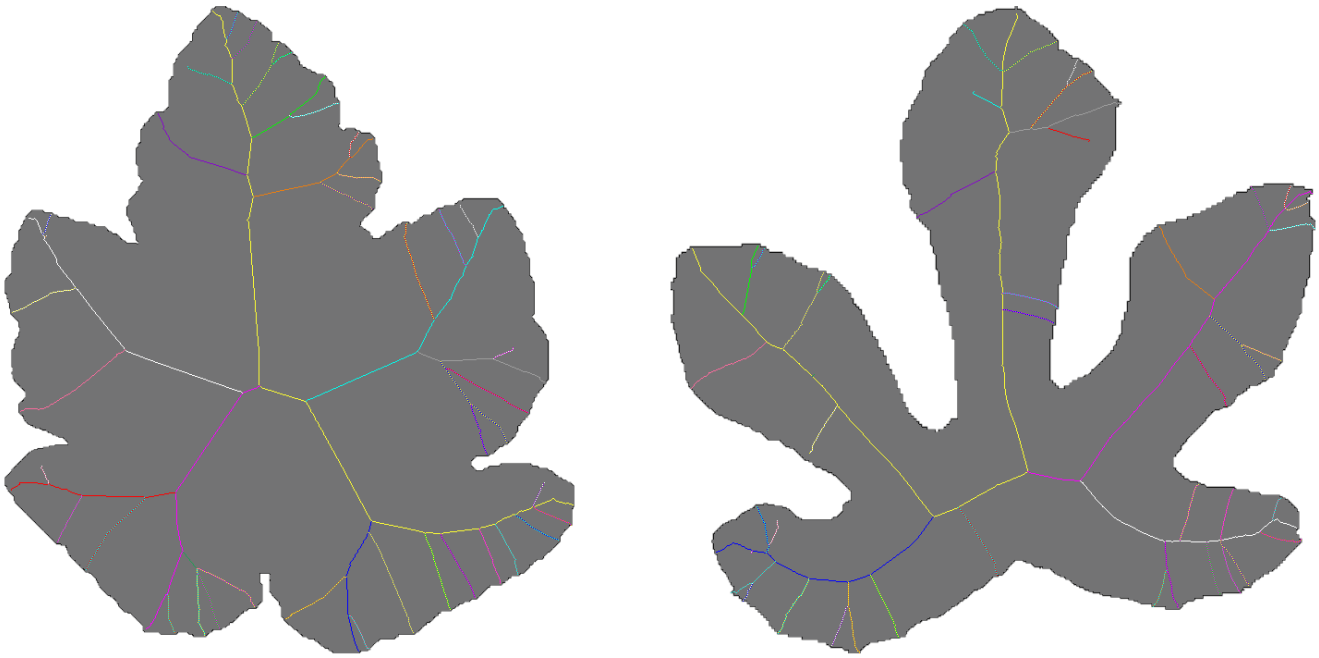


FIGURE 5.19 – *Appariement de deux feuilles d'arbres provenant de la même espèce (carica) et ayant des formes très différentes. La valeur de dissimilarité dans ce cas est 932.*

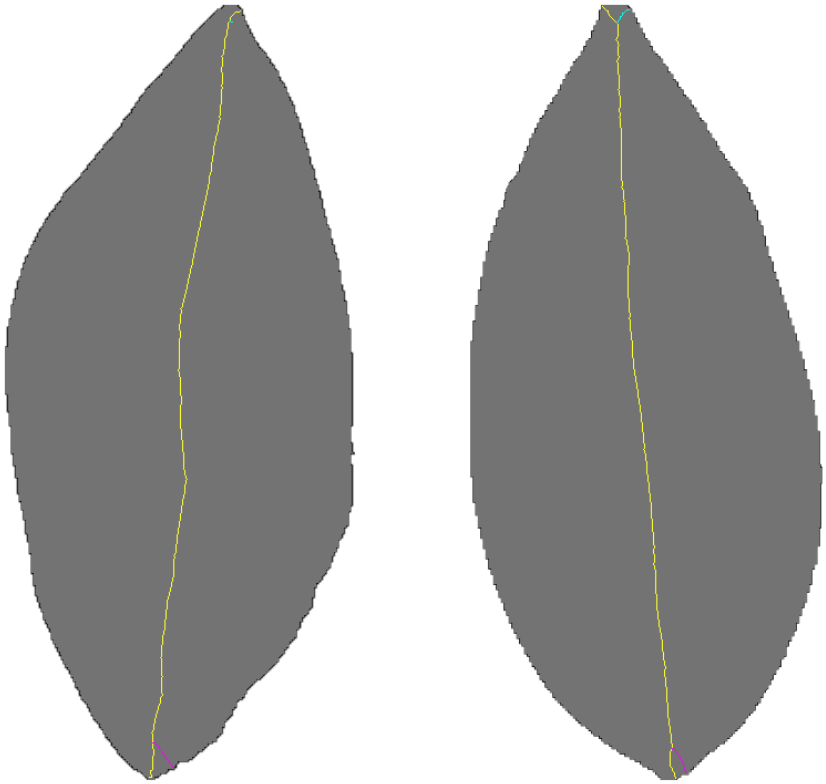
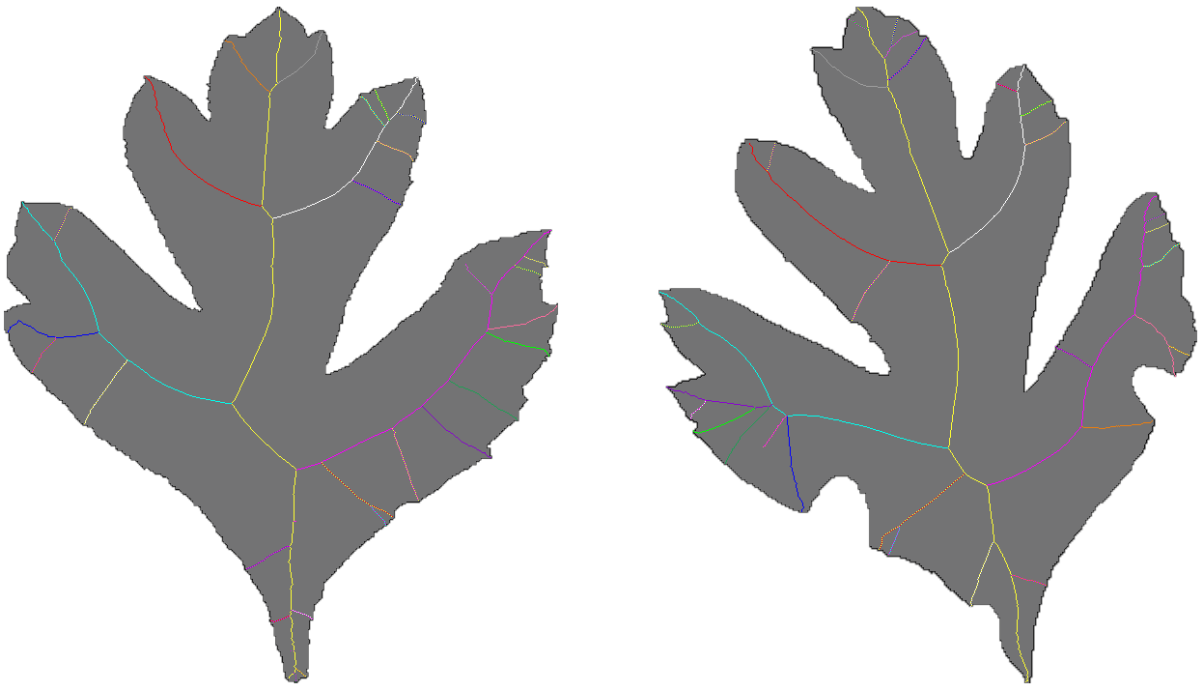
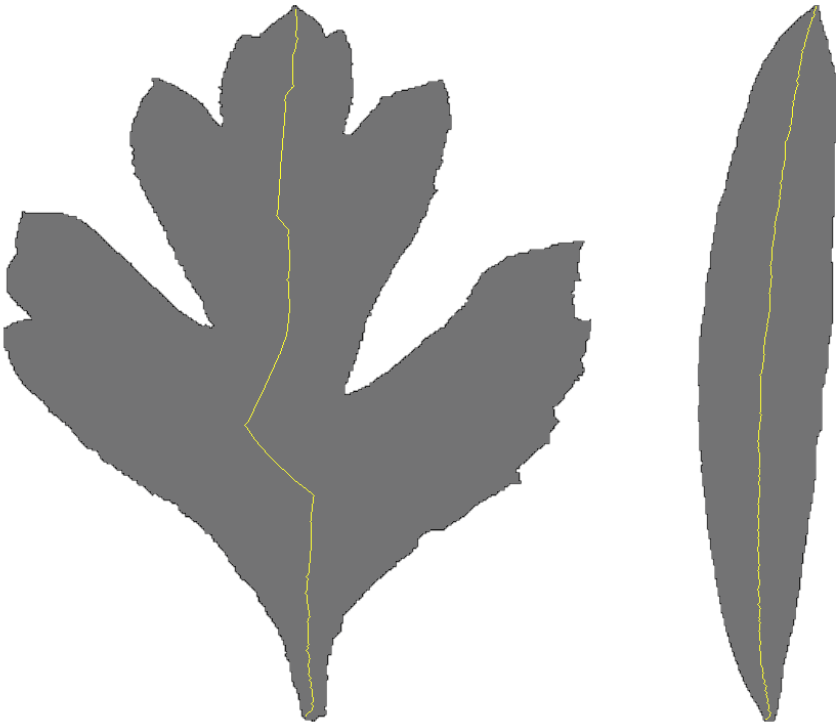


FIGURE 5.20 – *Appariement de deux feuilles d'arbres provenant de deux espèces différentes (grandiflora et aquifolium) mais ayant des formes très proches. La valeur de dissimilarité dans ce cas est 116.*

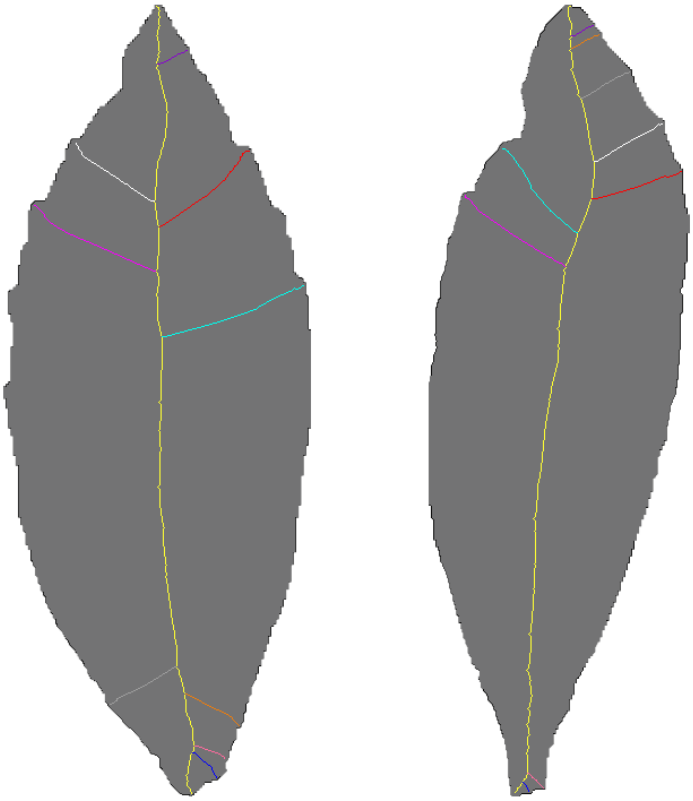
5.2.5 Exemples d'appariements de feuilles d'arbres



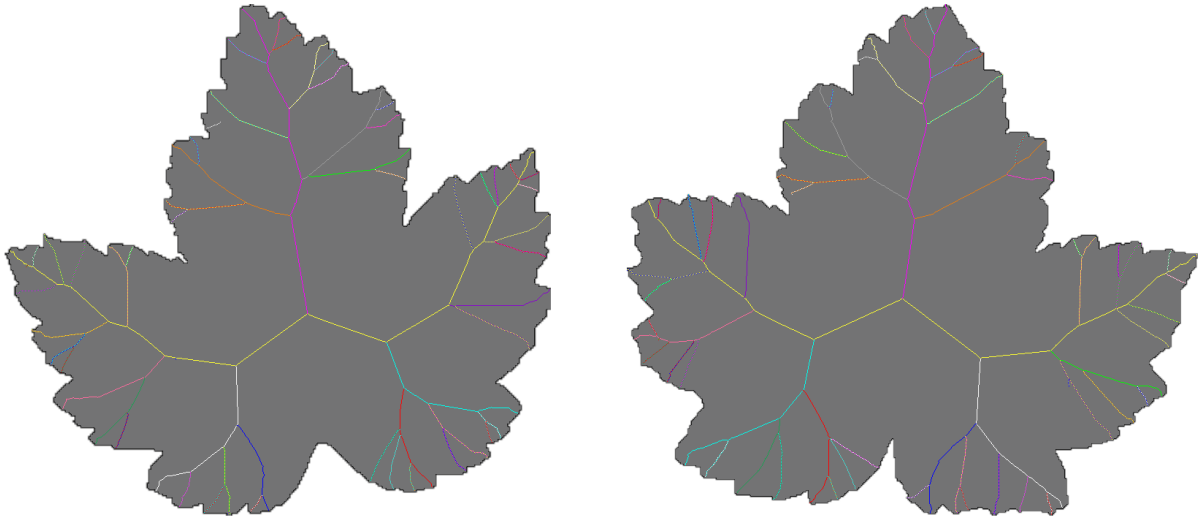
Azarolus/Azarolus : dissimilarité=122



Azarolus/Europaea : dissimilarité=803



Japonica/Japonica : dissimilarité=152



Platanoide/Platanoide : dissimilarité=108

Conclusion du chapitre

Dans ce chapitre, qui traite de la dernière étape à réaliser pour reconnaître une forme, nous avons présenté notre algorithme d'appariement d'hyper-graphes.

Le but est d'apparier les hyper-sommets correspondant aux méta-branches décrites au Chapitre 4. Pour ce faire, nous nous basons sur les caractéristiques locales des hyper-sommets (longueur et importance des méta-branches, ainsi que la fonction de distance le long de ces méta-branches) et sur le contexte (plus court chemin entre chaque paire d'hyper-sommets). Nous utilisons la distance du cantonnier pour comparer ces différentes caractéristiques dans le but de trouver la dissimilarité entre deux hyper-sommets. Ensuite, nous associons les hyper-sommets deux à deux grâce à l'algorithme hongrois. La dernière étape est un raffinement itératif de l'appariement avec mise à jour des coûts d'appariement. Ce processus de raffinement est itéré jusqu'à vérification d'un critère de stabilité.

La deuxième partie de ce chapitre expose les résultats que nous avons obtenus. Nous avons notamment testé l'influence des différents paramètres sur les performances, pour justifier les valeurs utilisées. Nous avons aussi comparé les performances de notre méthode à celles de deux autres méthodes existantes et avons étudié la complexité de notre méthode en regard de celle d'une des méthodes concurrentes, également basée sur des distances entres branches (algorithme de Bai et Latecki (2008)). À notre sens, l'inconvénient de cette dernière méthode est sa complexité algorithmique trop élevée. Bien que nous obtenions des taux de reconnaissance un peu en-deçà de ceux de Bai et Latecki, la complexité de notre méthode est moindre.

Finalement, nous avons appliqué notre méthode sur une base de données de feuilles d'arbres et avons obtenu un taux de reconnaissance de 72%, ce qui est déjà très satisfaisant compte tenu de la difficulté de ce type de données. Nous avons prévu, prochainement, d'intégrer notre méthode dans le pipeline de reconnaissance développé dans le cadre de l'ANR ReVeS (Reconnaissance de Végétaux sur Smartphone).

Conclusion générale

Apports scientifiques et résumé

Au cours de ce mémoire, nous avons développé chacune des étapes permettant d'apparier des formes planes à partir de leur masque binaire.

Squelettisation

Nous avons commencé par extraire un squelette homotope à la forme, fin et robuste au bruit. Pour ce faire, nous avons conçu une méthode de squelettisation, nommée DECS (Leborgne et al. 2014a;b; 2015), basée sur une carte de distances. Elle consiste à effectuer une propagation sur les crêtes et centres de boules maximales à partir du centre de la boule maximale ayant le plus grand rayon. Puis, l'ensemble de pixels obtenu est aminci à l'aide d'un algorithme parallèle basé sur la suppression des points simples. Le squelette est enfin élagué afin d'éliminer les branches non-significatives. Nous avons comparé cet algorithme avec les algorithmes de la littérature qui nous ont paru les plus pertinents pour l'application visée et avons pu montrer l'intérêt de notre approche au travers des résultats.

Hiérarchisation

Les branches de squelette n'ont pas toutes la même contribution à la forme. Nous avons choisi de quantifier cette importance et de favoriser l'appariement des branches d'importance similaire, en attribuant plus de poids à celles qui codent l'allure générale de la forme et moins à celles provenant de détails. Pour y parvenir, nous lissons successivement la forme d'origine en conservant son aire et à chaque itération, nous déformons le squelette initialement calculé sur la forme d'origine afin qu'il reste en adéquation avec la forme lissée. Les branches de squelettes raccourcissent puis disparaissent au fur et à mesure du lissage. Nous avons quantifié l'importance d'une branche avec son degré de persistance au cours des lissages successifs. Ainsi, nous obtenons une hiérarchisation du squelette.

En outre, d'une part, nous avons montré que cette hiérarchisation n'était pas spécialement propre à DECS, mais qu'elle pouvait être appliquée sur n'importe quelle méthode de squelettisation pour parvenir à un élagage des squelettes afin de les rendre robustes au bruit. D'autre part, nous avons modélisé les squelettes hiérarchiques par des hyper-graphes hiérarchiques pour obtenir une structure de données plus facilement manipulable lors de l'appariement.

Appariement

Finalement, nous avons élaboré une méthode permettant d'apparier nos hyper-graphes hiérarchiques de manière à déterminer la forme la plus proche d'une *forme requête* dans une base de données. Notre méthode est basée, d'une part, sur les caractéristiques intrinsèques des branches du squelette hiérarchique (méta-branches), et d'autre part, sur le contexte, c'est-à-dire sur leurs positions relatives par rapport aux autres méta-branches. Là encore, nous avons comparé notre proposition avec les algorithmes de la littérature qui nous ont paru les plus efficaces. Nous avons ici montré, d'une part, que les résultats obtenus étaient comparables avec ceux de la littérature mais avec un algorithme de complexité moindre. D'autre part, nous avons utilisé notre proposition dans un contexte réel pour reconnaître des feuilles d'arbre, contexte somme toute difficile compte tenu de la grande variabilité intra-classe et de la faible variabilité inter-classe. Les résultats obtenus dans ce cadre sont en adéquation avec ceux de la littérature et laissent présager qu'il pourrait y avoir un intérêt à les combiner avec ceux d'autres descripteurs.

Extension du travail de thèse

Lors d'une collaboration avec Antoine VACAVANT (Maître de Conférences à l'IUT du Puy-en-Velay), nous avons travaillé sur l'obtention d'un graphe de Reeb (Reeb 1946) en deux dimensions robuste au bruit à partir du squelette obtenu avec la méthode DECS. À noter que ce travail ne faisant pas réellement partie des travaux de thèse, nous n'en avons pas fait état dans le manuscrit. Cette collaboration a donné lieu à un article que nous avons présenté le mois dernier lors de la conférence internationale CTIC à Marseille (Vacavant et Leborgne (2016)).

Le graphe de Reeb est un objet mathématique permettant de décrire la topologie d'une forme. Il est basé sur une fonction de hauteur (ou fonction de Morse) h définie sur $\mathcal{F}$ dans notre cas. Chaque arête du graphe de Reeb est composée des points appartenant à la même composante connexe. Les nœuds représentent les points critiques de la fonction h qui sont :

- *begin* : valeurs minimales de h
- *end* : valeurs maximales de h
- *split* : endroit où deux arêtes se divisent lors de la construction
- *merge* : endroit où deux arêtes fusionnent lors de la construction

Pour ces deux derniers points critiques, il s'agit des valeurs sur les points selles de h. La Figure 5.21 montre un exemple de graphe de Reeb simple. L'intérêt du graphe de Reeb est qu'il a une structure plus légère que nos hyper-graphes du fait qu'il ne capture que la topologie de la forme. Il est donc plus facile à manipuler mais moins précis.

Pour créer un graphe de Reeb à partir d'un squelette obtenu grâce à la méthode DECS, nous parcourons ce dernier en largeur, à partir d'un point fixé au départ, étiqueté *begin*. Chaque extrémité de branche est alors un point critique. La Figure 5.22 représente un exemple de squelette obtenu avec DECS sur une forme très bruitée, et son graphe de Reeb associé.

Ce travail mériterait d'être poursuivi et notamment utilisé dans le cadre de l'appariement.

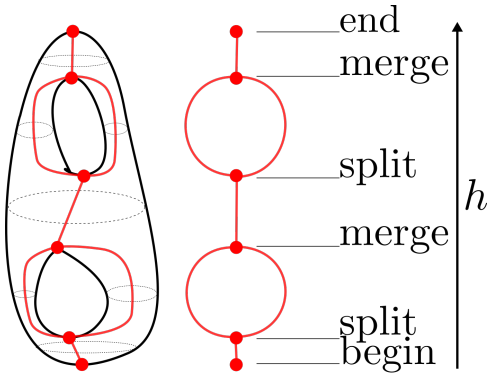


FIGURE 5.21 – *Notations et illustration d'un graphe de Reeb sur une forme simple.*

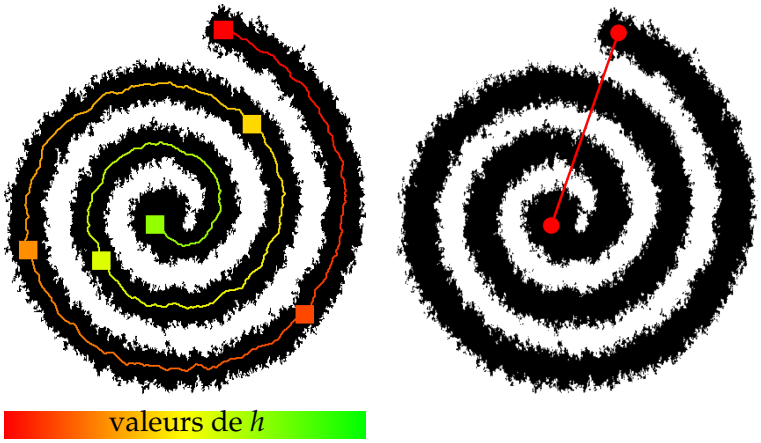


FIGURE 5.22 – *Squelette d'une spirale bruitée et son graphe de Reeb associé. Les pixels du squelette sont coloriés en fonction de leur valeur de h (voir palette) et les carrés coloriés de la même couleur représentent l'ensemble des pixels ayant la même valeur.*

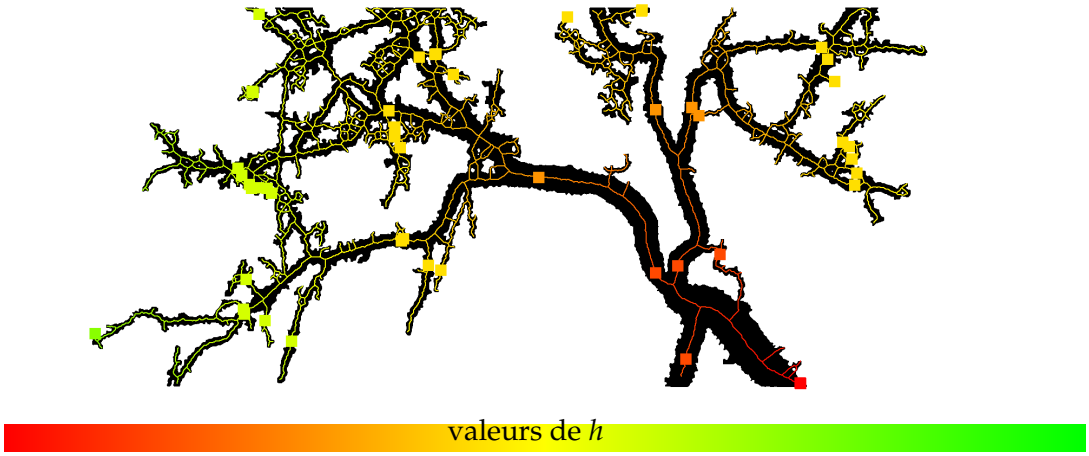
Nous pourrions alors le comparer à notre travail de thèse. À noter également qu'il pourrait être utilisé pour étudier, par exemple, des vaisseaux dans les images médicales obtenues par angiographie. Un exemple est présenté sur la Figure 5.23.

PERSPECTIVES

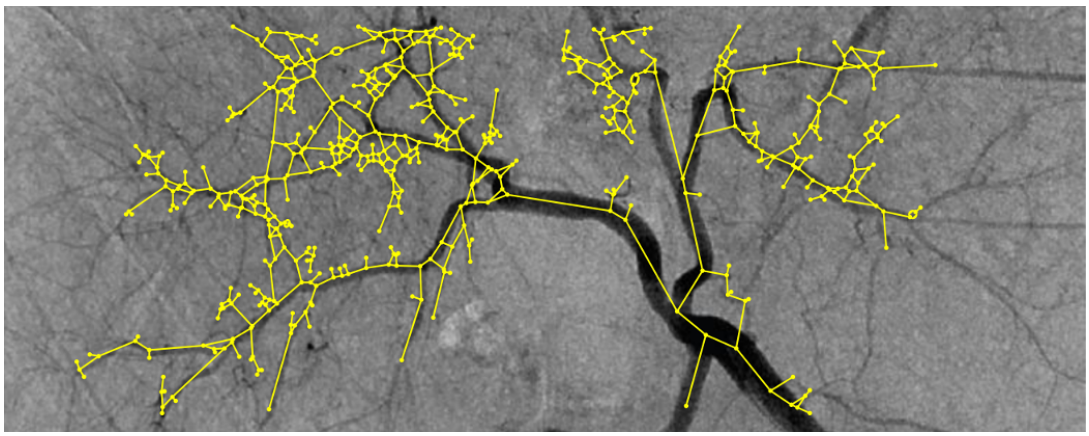
Extension à des topologies plus complexes et à la 3D

Dans la continuité directe de notre travail de thèse, nous pouvons étendre la création de squelettes hiérarchiques et leurs appariements à des formes trouées. Pour cela, il faudrait s'assurer que les formes lissées restent homotopes à la forme initiale. Il faudrait également généraliser la condition d'arrêt du lissage, c'est-à-dire déterminer la structure minimale du squelette en fonction du nombre de trous de la forme (l'utilisation du graphe de Reeb permet de déterminer ce dernier facilement). Contrairement aux travaux réalisés dans la thèse, la structure minimale n'est pas une méta-branche mais un ensemble de méta-branches, comme le montre la Figure 5.24. Lors de l'étape d'appariement, il faudrait être capable soit d'apparier des ensembles de branches, soit de déterminer un moyen de partitionner ces branches.

Dans le cadre de cette thèse, nous nous sommes focalisés sur des formes en deux dimen-



(a)



(b)

FIGURE 5.23 – *Extraction du graphe de Reeb à partir d'une image d'angiographie. (a) Squelette des vaisseaux sanguins sur une image d'angiographie segmentée. Les pixels du squelette sont coloriés en fonction de leur valeur de h (voir palette) et les carrés coloriés de la même couleur représentent l'ensemble des pixels ayant la même valeur. (b) Graphe de Reeb associé superposé à l'image initiale.*

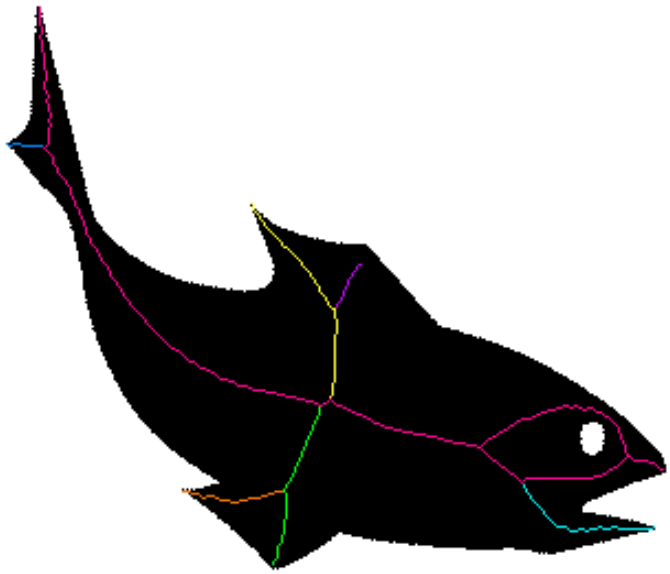


FIGURE 5.24 – *Travaux préliminaires mettant en évidence la structure minimale du squelette, en rose, comme un ensemble de méta-branches lorsqu'une forme présente un trou.*

sions puisqu'une application première était la reconnaisance de feuilles d'arbres. Néanmoins, la méthode DECS pourrait être étendue à une dimension supérieure. Les applications seraient alors multipliées. Nous pourrions, par exemple, traiter des images d'IRM/Scanner pour caractériser la forme des vaisseaux mais aussi faire de la reconstruction ou de la représentation de formes en créant des atlas de réseaux vasculaires en 3 dimensions. Grâce à ce squelette en trois dimensions, il serait également possible de guider les algorithmes de segmentation d'images médicales (Karmakar et al. 2015), ou de recaler ces images (Osorio et al. 2012).

Appariement

Nous pensons qu'il manque quelques tests à réaliser, pour l'appariement notamment. Par faute de temps et de disponibilité de code, nous n'avons pas pu les effectuer. Il convient de tester chaque étape et la comparer à celles de la littérature effectuant la même tâche.

Nous envisageons de comparer notre travail avec d'autres méthodes basées squelette comme les graphes d'os (Macrini et al. 2008; 2011a;b), mais également avec des méthodes utilisant un tout autre descripteur de formes que le squelette, comme les méthodes récentes de contexte de formes (Ling et Jacobs 2007, Xie et al. 2008).

Une application directe et très intéressante du travail que nous avons réalisé serait de calculer des moyennes de formes et de squelettes hiérarchiques à partir de l'appariement de leurs méta-branches en y intégrant des statistiques sur la longueur et/ou l'épaisseur de la forme associée à chaque méta-branche pour obtenir une caractérisation et une variabilité du squelette moyen. Cet apport serait potentiellement une avancée par rapport à l'état de l'art actuel. L'uti-

lisation des statistiques se justifie car deux formes pouvant faire partie de la même catégorie peuvent présenter quelques différences géométriques, comme nous l'avons vu durant cette thèse. L'application pourrait être de trois types :

1. Requête : Trouver la similarité entre une *forme requête* et le squelette moyen d'une catégorie de formes. L'utilisation du squelette moyen permettrait donc de réduire le nombre de formes à comparer.
2. Classification : Créer des classes dans une base de données en associant les formes ayant des caractéristiques communes puis faire évoluer le squelette moyen de chaque catégorie en fonction des formes ajoutées dans chacune d'elles.
3. Atlas : Recaler des formes que nous savons similaires, c'est-à-dire les amener dans le même espace en tenant compte de la variabilité qu'il peut y avoir entre elles.

BIBLIOGRAPHIE

Tomasz Adamek et Noel E O'Connor. A multiscale representation method for nonrigid shapes with a single closed contour. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5) :742–753, 2004. (Cité page 34.)

Maher Ahmed et Rabab Ward. A rotation invariant rule-based thinning algorithm for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12) :1672–1678, 2002. (Cité page 40.)

Naif Alajlan, Ibrahim El Rube, Mohamed S Kamel, et George Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*, 40(7) :1911–1920, 2007. (Cité page 34.)

Carlo Arcelli et Gabriella Sanniti di Baja. A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4) :411–414, 1989. (Cité page 41.)

Carlo Arcelli et Gabriella Sanniti di Baja. Ridge points in euclidean distance maps. *Pattern Recognition Letters*, 13(4) :237–243, 1992. (Cité page 42.)

Carlo Arcelli et Gabriella Sanniti di Baja. Euclidean skeleton via centre-of-maximal-disc extraction. *Image and Vision Computing*, 11(3) :163–173, 1993. (Cité pages 42 et 46.)

Dominique Attali, Gabriella Sanniti di Baja, et Edouard Thiel. Pruning discrete and semicontinuous skeletons. Dans *Image Analysis and Processing*, pages 488–493, 1995. (Cité page 53.)

Dominique Attali et Edouard Thiel. Du squelette discret ou continu, 1993. (Cité page 41.)

Xiang Bai et Longin Jan Latecki. Discrete skeleton evolution. Dans *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 362–374, 2007. (Cité page 53.)

Xiang Bai et Longin Jan Latecki. Path similarity skeleton graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7) :1282–1292, 2008. (Cité pages xvi, 59, 118, 129, 130, 144, 145, 146, 153, 154 et 159.)

Xiang Bai, Longin Jan Latecki, et Wen-Yu Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3) : 449–462, 2007. (Cité pages 51, 53 et 55.)

Xiang Bai, Xingwei Yang, Longin Jan Latecki, Wenyu Liu, et Zhuowen Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5) :861–874, 2010. (Cité page 62.)

Emre Baseski, Aykut Erdem, et Sibel Tari. Dissimilarity between two skeletal trees in a context. *Pattern Recognition*, 42(3) :370–385, 2009. (Cité page 62.)

Serge Belongie, Jitendra Malik, et Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4) :509–522, 2002. (Cité pages xvi, 34, 145, 146, 153 et 154.)

Thierry M Bernard et Antoine Manzanera. Improved low complexity fully parallel thinning algorithm. Dans *Image Analysis and Processing*, pages 215–220, 1999. (Cité pages 75, 89 et 97.)

Stefano Berretti, Alberto Del Bimbo, et Pietro Pala. Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Multimedia*, 2(4) :225–239, 2000. (Cité page 34.)

Gilles Bertrand et Michel Couprie. Powerful parallel and symmetric 3d thinning schemes based on critical kernels. *Journal of Mathematical Imaging and Vision*, pages 1–15, 2014. (Cité pages viii, xiii, xiv, xvi, 40, 65, 67, 77, 78, 82, 84, 88, 90, 121 et 122.)

Bir Bhanu et Xuejun Tan. *Computational Algorithms for Fingerprint Recognition*, volume 1. Springer Science and Business Media, 2012. (Cité page 2.)

Harry Blum. A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form*, 1967. (Cité pages 38 et 47.)

Miroslaw Bober. Mpeg-7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6) :716–719, 2001. (Cité page 35.)

Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3) :344–371, 1986. (Cité page 41.)

Gunilla Borgefors, Giuliana Ramella, et Gabriella Sanniti di Baja. Hierarchical decomposition of multiscale skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11) : 1296–1312, 2001. (Cité pages xii, 51, 52 et 55.)

Gunilla Borgefors et Gabriella Sanniti di Baja. Skeletonizing the distance transform on the hexagonal grid. Dans *Pattern Recognition*, pages 504–507, 1988. (Cité page 41.)

Jonathan W Brandt et V Ralph Algazi. Continuous skeleton computation by voronoi diagram. *Computer Vision, Graphics, and Image Processing*, 55(3) :329–338, 1992. (Cité page 44.)

J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM System Journal*, 4 : 25–30, 1965. (Cité page 18.)

Guillaume Cerutti. *Segmentation et interprétation d'images naturelles pour l'identification de feuilles d'arbres sur smartphone*. Thèse de doctorat en informatique, Université Lumière Lyon 2, 2013. URL `http://liris.cnrs.fr/publis/?id=6375`. (Cité pages 3 et 155.)

Sukmoon Chang. Extracting skeletons from distance maps. *International Journal of Computer Science and Network Security*, 7(7) :213–219, 2007. (Cité page 41.)

Houssem Chatbri, Keisuke Kameyama, et Paul Kwan. A comparative study using contours and skeletons as shape representations for binary image matching. *Pattern Recognition Letters*, 2015. (Cité page 34.)

John Chaussard, Michel Couprie, et Hugues Talbot. Robust skeletonization using the discrete λ-medial axis. *Pattern Recognition Letters*, 32(9) :1384–1394, 2011. (Cité page 52.)

Frédéric Chazal et André Lieutier. The λ-medial axis. *Graphical Models*, 67(4) :304–331, 2005. (Cité page 52.)

Chaur-Chin Chen. Improved moment invariants for shape discrimination. *Pattern recognition*, 26(5) :683–686, 1993. (Cité page 34.)

Jing-Chao Chen. Dijkstra's shortest path algorithm. *Journal of Formalized Mathematics*, 15 : 144–157, 2003. (Cité pages 133 et 134.)

Minsu Cho, Jungmin Lee, et Kyoung Mu Lee. Reweighted random walks for graph matching. Dans *European Conference on Computer Vision*, pages 492–505. Springer, 2010. (Cité page 60.)

Wai-Pak Choi, Kin-Man Lam, et Wan-Chi Siu. Extraction of the euclidean skeleton based on a connectivity criterion. *Pattern Recognition*, 36(3) :721–729, 2003. (Cité pages viii, xiii, xiv, 41, 42, 65, 67, 78, 79, 82, 83, 84, 86, 88, 90, 92 et 93.)

David Coeurjolly et Annick Montanvert. Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3) :437–448, 2007. (Cité pages xi, xii, xiii, 21, 28, 29, 41, 42, 69, 70, 71, 84 et 88.)

David Coeurjolly, Annick Montanvert, et Jean-Marc Chassery. *Géométrie discrète et images numériques*. Hermès, 2007. (Cité page 29.)

Michel Couprie, David Coeurjolly, et Rita Zrour. Discrete bisector function and euclidean skeleton in 2d and 3d. *Image and Vision Computing*, 25(10) :1543–1556, 2007. (Cité page 46.)

Tim Culver. *Computing the Medial Axis of a Polyhedron Reliably and Efficiently*. The University of North Carolina at Chapel Hill, 2000. (Cité page 44.)

P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3) : 227–248, 1980. (Cité page 20.)

Julien Dardenne, Sébastien Valette, Nicolas Siauve, et Rémy Prost. Approximation de l'axe médian pour les objets discrets avec prise en compte de la courbure. Dans *XXIIe colloque GRETSI (Groupe d'Etudes du Traitement du Signal et des Images)*, 2009. (Cité pages xii et 44.)

Amitava Datta et Swapan K Parui. A robust parallel thinning algorithm for binary images. *Pattern recognition*, 27(9) :1181–1192, 1994. (Cité page 39.)

E Roy Davies. *Machine Vision : Theory, Algorithms, Practicalities*. Elsevier, 2004. (Cité page 34.)

Isabelle Debled-Rennesson. *Etude et reconnaissance des droites et plans discrets*. PhD thesis, 1995. (Cité page 34.)

M Fatih Demirci, Ali Shokoufandeh, Yakov Keselman, Lars Bretzner, et Sven Dickinson. Object recognition as many-to-many feature matching. *International Journal of Computer Vision*, 69 (2) :203–222, 2006. (Cité pages xiii et 59.)

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1) :269–271, 1959. (Cité pages 133 et 134.)

Italo Capuzzo Dolcetta, Stefano Finzi Vita, et Ricardo March. Area preserving curve shortening flows : from phase transitions to image processing. *Interfaces and Free Boundaries*, 4(4) :325–344, 2002. (Cité pages viii, 99 et 103.)

Aykut Erdem et Sibel Tari. A similarity-based approach for shape classification using aslan skeletons. *Pattern Recognition Letters*, 31(13) :2024–2032, 2010. (Cité page 62.)

Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, et Horst Bunke. Approximation of graph edit distance based on hausdorff matching. *Pattern Recognition*, 48(2) :331–343, 2015. (Cité page 57.)

Herbert Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, (2) :260–268, 1961. (Cité pages 15 et 34.)

Michael Gage. On an area-preserving evolution equation for plane curves. *Contemporary Mathematics*, 51 :51–62, 1986. (Cité pages viii, 99, 103 et 104.)

Xinbo Gao, Bing Xiao, Dacheng Tao, et Xuelong Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1) :113–129, 2010. (Cité page 57.)

Haythem Ghazouani. *Navigation Visuelle de Robots Mobiles dans un Environnement d'Intérieur*. PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc, 2012. (Cité page 2.)

Ngo Truong Giang, Ngo Quoc Tao, Nguyen Duc Dung, et Nguyen Trong The. Skeleton based shape matching using reweighted random walks. Dans *Information, Communications and Signal Processing*, pages 1–5, 2013a. (Cité page 60.)

Ngo Truong Giang, Ngo Quoc Taob, et Nguyen Duc Dung. Skeleton-based shape matching using higher-order constraints. *Journal of Pattern Recognition and Image Processing*, 4(4) :443–454, 2013b. (Cité pages xii, 56 et 60.)

Lena Gorelick, Meirav Galun, Eitan Sharon, Ronen Basri, et Achi Brandt. Shape representation and classification using the poisson equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12) :1991–2005, 2006. (Cité page 41.)

Ardeshir Goshtasby. Description and discrimination of planar shapes using shape matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6) :738–743, 1985. (Cité page 35.)

Hervé Goëau, Pierre Bonnet, Alexis Joly, Nozha Boujemaa, Daniel Barthélémy, Jean-François Molino, P. Birnbaum, E. Mouysset, et M. Picard. The image clef 2011 plant images classification task. Dans *CLEF*, 2011. (Cité page 155.)

Hervé Goëau, Pierre Bonnet, Alexis Joly, I. Yahiaoui, Daniel Barthélémy, Nozha Boujemaa, et Jean-François Molino. The image clef 2013 plant identification task. Dans *CLEF*, 2012. (Cité page 155.)

William I Groskey et Rajiv Mehrotra. Index-based object recognition in pictorial data management. *Computer vision, Graphics, and Image Processing*, 52(3) :416–436, 1990. (Cité page 34.)

William I Groskey, Peter Neo, et Rajiv Mehrotra. A pictorial index mechanism for model-based matching. *Data and Knowledge Engineering*, 8(4) :309–327, 1992. (Cité page 34.)

C. Judith Hilditch. Linear skeletons from square cupboards. *Machine Intelligence*, 4 :403–420, 1969. (Cité page 13.)

F.S. Hillier et G.J. Lieberman. *Introduction to Mathematical Programming*. Industrial Engineering and Management Science. McGraw-Hill, 1995. ISBN 9780070480278. (Cité page 137.)

Chia-Chun Hung, Eric T Carlson, et Charles E Connor. Medial axis shape coding in macaque inferotemporal cortex. *Neuron*, 74(6) :1099–1113, 2012. (Cité page 35.)

Nilanjana Karmakar, Arindam Biswas, et Partha Bhowmick. Reeb graph based segmentation of articulated components of 3d digital objects. *Theoretical Computer Science*, 2015. (Cité page 165.)

Michael Kass, Andrew Witkin, et Demetri Terzopoulos. Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, 1988. (Cité page 112.)

Yakov Keselman, Ali Shokoufandeh, M Fatih Demirci, et Sven Dickinson. Many-to-many graph matching via metric embedding. Dans *Computer Vision and Pattern Recognition*, volume 1, pages I–850, 2003. (Cité page 59.)

Duck Hoon Kim, Il Dong Yun, et Sang Uk Lee. Attributed relational graph matching based on the nested assignment structure. *Pattern Recognition*, 43(3) :914–928, 2010. (Cité pages 129, 140 et 147.)

Ron Kimmel, Doron Shaked, Nahum Kiryati, et Alfred M Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62(3) :382–391, 1995. (Cité pages 42 et 46.)

V.A. Kovalevsky. New definition and fast recognition of digital straight segments and arcs. *International Conference of Pattern Recognition*, 2 :31–34, 1990. (Cité page 8.)

V.A. Kovalevsky. Algorithms and data structures for computer topology. *Digital and Image Geometry*, pages 38–58, 2001. (Cité page 15.)

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2) :83–97, 1955. (Cité pages ix, 125, 139 et 147.)

Louisa Lam, Seong-Whan Lee, et Ching Y Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9) :869–885, 1992. (Cité page 40.)

Longin Jan Latecki, Rolf Lakämper, et Ulrich Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. Dans *Computer Vision and Pattern Recognition*, volume 1, pages 424–429, 2000. (Cité pages xiv, 81, 84 et 87.)

Longin Jan Latecki, Quan-nan Li, Xiang Bai, et Wen-yu Liu. Skeletonization using ssm of the distance transform. Dans *International Conference on Image Processing*, volume 5, pages V–349, 2007a. (Cité page 42.)

Longin Jan Latecki, Qiang Wang, Suzan Koknar-Tezel, et Vasileios Megalooikonomou. Optimal subsequence bijection. Dans *International Conference on Data Mining*, pages 565–570, 2007b. (Cité page 145.)

Eugene L Lawler. *Combinatorial Optimization : Networks and Matroids*. New York : Holt, Rinehart and Winston, 2001. (Cité page 147.)

Aurélie Leborgne, Julien Mille, et Laure Tougne. Noise-resistant digital euclidean connected skeleton for graph-based shape matching. *Journal of Visual Communication and Image Representation*, 31 :165–176, 2015. (Cité pages xiv, 3, 82, 97, 101, 106, 115, 119 et 161.)

Aurélie Leborgne, Julien Mille, et Laure Tougne. Extracting noise-resistant skeleton on digital shapes for graph matching. Dans *International Symposium on Visual Computing*, pages 293–302, 2014a. (Cité pages 3, 97 et 161.)

Aurélie Leborgne, Julien Mille, et Laure Tougne. Squelette Euclidien Discret Connecté (DECS) Résistant au Bruit pour l'Appariement de Formes Basé Graphes. Dans *Compression et Représentation des Signaux Audiovisuels (CORESA)*, 2014b. (Cité pages 3, 97 et 161.)

Aurélie Leborgne, Julien Mille, et Laure Tougne. Hierarchical skeleton for shape matching. Dans *International Conference on Image Processing*, 2016a. (Cité page 3.)

Aurélie Leborgne, Julien Mille, et Laure Tougne. Squelette hiérarchique pour la description de formes. Dans *Reconnaissance de Formes et Intelligence Artificielle*, 2016b. (Cité page 3.)

Frank Lebourgeois et Hubert Emptoz. Skeletonization by gradient regularization and diffusion. Dans *IEEE International Conference on Image Processing*, volume 2, pages 1118–1122, 2007. (Cité page 41.)

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Dans *Soviet Physics Doklady*, volume 10, pages 707–710, 1966. (Cité page 56.)

Frédéric Leymarie et Martin D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1) :56–75, 1992. (Cité page 38.)

Frédéric Leymarie et Martin D. Levine. Snakes and skeletons. *Int Rept*, 1101 :89–3, 1989. (Cité page 38.)

Michael Leyton. Symmetry-curvature duality. *Computer Vision, Graphics, and Image Processing*, 38(3) :327–341, 1987. (Cité page 67.)

Laurence Likforman-Sulem et Elisa Barney-Smith. *IMAGERIE - Reconnaissance des formes - Théorie et pratique sous Matlab - Cours et exercices corrigés (niveau C)*. 2013. (Cité pages xi et 2.)

Haibin Ling et David W Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2) :286–299, 2007. (Cité pages 34 et 165.)

HongZhi Liu, Zhong-Hai Wu, Xing Zhang, et D Frank Hsu. A skeleton pruning algorithm based on information fusion. *Pattern Recognition Letters*, 34(10) :1138–1145, 2013. (Cité pages 53 et 55.)

Hongzhi Liu, Zhonghai Wu, D. Frank Hsu, Bradley S Peterson, et Dongrong Xu. On the generation and pruning of skeletons using generalized voronoi diagrams. *Pattern Recognition Letters*, 33(16) :2113–2119, 2012. (Cité pages 53 et 55.)

Tyng-Luh Liu et Davi Geiger. Approximate tree matching and shape similarity. Dans *International Conference on Computer Vision*, volume 1, pages 456–462, 1999. (Cité page 62.)

Lorenzo Livi et Antonello Rizzi. The graph matching problem. *Pattern Analysis and Applications*, 16(3) :253–283, 2013. (Cité page 56.)

Christophe Lohou et Julien Dehos. Automatic correction of ma and sonka's thinning algorithm using p-simple points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6) : 1148–1152, 2010. (Cité page 40.)

Sven Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8) :983–1001, 1998. (Cité page 35.)

Diego Macrini, Sven Dickinson, David Fleet, et Kaleem Siddiqi. Bone graphs : Medial shape parsing and abstraction. *Computer Vision and Image Understanding*, 115(7) :1044–1061, 2011a. (Cité pages 50, 58 et 165.)

Diego Macrini, Sven Dickinson, David Fleet, et Kaleem Siddiqi. Object categorization using bone graphs. *Computer Vision and Image Understanding*, 115(8) :1187–1206, 2011b. (Cité pages 50, 58 et 165.)

Diego Macrini, Kaleem Siddiqi, et Sven Dickinson. From skeletons to bone graphs : Medial abstraction for object recognition. Dans *Computer Vision and Pattern Recognition*, pages 1–8, 2008. (Cité pages 50, 58 et 165.)

Grégoire Malandain et Sara Fernández-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16(5) :317–327, 1998. (Cité page 41.)

Davide Maltoni, Dario Maio, Anil Jain, et Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer Science and Business Media, 2009. (Cité page 2.)

Antoine Manzanera, Thierry M Bernard, Bernard Longuet, et al. N-dimensional skeletonization : a unified mathematical framework. *Journal of Electronic Imaging*, 11(1) :25–37, 2002. (Cité page 39.)

Antoine Manzanera, Thierry M Bernard, Françoise Prêteux, et Bernard Longuet. Ultra-fast skeleton based on an isotropic fully parallel algorithm. Dans *Discrete Geometry for Computer Imagery*, pages 313–324, 1999. (Cité pages 75 et 97.)

Niranjan Mayya et VT Rajan. Voronoi diagrams of polygons : A framework for shape representation. *Journal of Mathematical Imaging and Vision*, 6(4) :355–378, 1996. (Cité page 44.)

Arnold Meijster, Jos BTM Roerdink, et Wim H Hesselink. A general algorithm for computing distance transforms in linear time. Dans *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 331–340. Springer, 2002. (Cité pages xi, 21, 41, 69 et 84.)

Djamel Merad, M Mallem, et S Lelandais. Skeletonization of two-dimensional regions using hybrid method. 2007. (Cité page 46.)

Victor Milenkovic. Robust construction of the voronoi diagram of a polyhedron. Dans *Canadian Conference on Computational Geometry*, volume 93, pages 473–478, 1993. (Cité page 44.)

Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5) :539–544, 1995. (Cité page 34.)

Farzin Mokhtarian, Sadegh Abbasi, Josef Kittler, et al. Efficient and robust retrieval by shape content through curvature scale space. *Series on Software Engineering and Knowledge Engineering*, 8 :51–58, 1997. (Cité page 34.)

Farzin Mokhtarian et Alan K Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (8) :789–805, 1992. (Cité pages 34 et 54.)

Ugo Montanari. A method for obtaining skeletons using a quasi-euclidean distance. *Journal of the Association for Computing Machinery*, 15(4) :600–624, 1968. (Cité page 41.)

Andrés Solís Montero et Jochen Lang. Skeleton pruning by contour approximation and the integer medial axis transform. *Computers and Graphics*, 36(5) :477–487, 2012. (Cité pages 53 et 55.)

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1) :32–38, 1957. (Cité page 140.)

C Wayne Niblack, Phillip B Gibbons, et David W Capson. Generating skeletons and centerlines from the distance transform. *Graphical Models and Image Processing*, 54(5) :420–437, 1992. (Cité page 42.)

R Ogniewicz et M Ilg. Voronoi skeletons : Theory and applications. Dans *Computer Vision and Pattern Recognition*, pages 63–69, 1992. (Cité page 44.)

Robert L Ogniewicz et Olaf Kübler. Hierarchic voronoi skeletons. *Pattern recognition*, 28(3) : 343–359, 1995. (Cité pages xii, 44, 47 et 49.)

Eliana M Vásquez Osorio, Mischa S Hoogeman, Alejandra Méndez Romero, Piotr Wielopolski, András Zolnay, et Ben JM Heijmen. Accurate ct/mr vessel-guided nonrigid registration of largely deformed livers. *Medical physics*, 39(5) :2463–2477, 2012. (Cité page 165.)

Kálmán Palágyi. Equivalent 2d sequential and parallel thinning algorithms. Dans *Combinatorial Image Analysis*, pages 91–100. Springer, 2014. (Cité pages 39 et 75.)

Nikos Paragios, James Duncan, et Nicholas Ayache. *Handbook of Biomedical Imaging*. Springer, 2008. (Cité page 2.)

Euripides GM Petrakis, Aristeidis Diplaros, et Evangelos Milios. Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11) :1501–1516, 2002. (Cité page 35.)

Markus Peura et Jukka Iivarinen. Efficiency of simple shape descriptors. *Aspects of Visual Form*, pages 443–451, 1997. (Cité page 34.)

Stephen M Pizer, William R Oliver, et Sandra H Bloomberg. Hierarchical shape description via the multiresolution symmetric axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4) :505–511, 1987. (Cité pages 46 et 47.)

Ingemar Ragnemalm. The euclidean distance transform in arbitrary dimensions. *Pattern Recognition Letters*, 14(11) :883–888, 1993. (Cité page 20.)

Georges Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. *CR Académie des Sciences de Paris*, 222(847-849) :2, 1946. (Cité pages 35 et 162.)

Peter Rockett. An improved rotation-invariant thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10) :1671–1674, 2005. (Cité page 40.)

A. Rosenfeld. Connectivity in digital pictures. *Journal of the Association for Computing Machinery*, 17(1) :146–160, 1970. (Cité pages 7 et 15.)

Azriel Rosenfeld. Arcs and curves in digital pictures. *Journal of the Association for Computing Machinery*, 20(1) :81–87, 1973. (Cité page 10.)

Azriel Rosenfeld et John L Pfaltz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13(4) :471–494, 1966. (Cité page 7.)

Caroline Rougier, Jean Meunier, Alain St-Arnaud, et Jacqueline Rousseau. Fall detection from human shape and motion history using video surveillance. Dans *Advanced Information Networking and Applications*, volume 2, pages 875–880, 2007. (Cité page 2.)

T. Roussillon. *Algorithmes d'Extraction de Modèles Géométriques Discrets pour la Représentation Robuste des Formes*. Thèse de doctorat en informatique, Université Lumière Lyon 2, 2009. URL `http://liris.cnrs.fr/publis/?id=4517`. (Cité page 15.)

D. Rutovitz. Pattern recognition. *Journal of Royal Statistical Society*, 129 :504–530, 1966. (Cité page 13.)

Khalid Saeed, Mariusz Rybnik, et Marek Tabedzki. Implementation and advanced results on the non-interrupted skeletonization algorithm. Dans *Computer Analysis of Images and Patterns*, pages 601–609, 2001. (Cité page 39.)

Khalid Saeed, Marek Tabędzki, Mariusz Rybnik, et Marcin Adamski. K3m : A universal algorithm for image skeletonization and a review of thinning techniques. *International Journal of Applied Mathematics and Computer Science*, 20(2) :317–335, 2010. (Cité page 39.)

Punam K Saha, Gunilla Borgefors, et Gabriella Sanniti di Baja. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 2015. (Cité page 45.)

Alberto Sanfeliu et King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, (3) :353–362, 1983. (Cité page 57.)

Gabriella Sanniti di Baja et Edouard Thiel. Skeletonization algorithm running on path-based distance maps. *Image and vision Computing*, 14(1) :47–57, 1996. (Cité pages 42 et 46.)

Lambert Schomaker, Edward de Leau, et Louis Vuurpijl. Using pen-based outlines for object-based annotation and image-based queries. Dans *Visual Information and Information Systems*, pages 585–592, 1999. (Cité page 1.)

Thomas B Sebastian et Benjamin B Kimia. Curves vs. skeletons in object recognition. *Signal Processing*, 85(2) :247–263, 2005. (Cité page 35.)

Thomas B Sebastian, Philip N Klein, et Benjamin B Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5) :550–571, 2004. (Cité pages xvi, 47, 57, 148 et 154.)

Luca Serino et Gabriella Sanniti di Baja. A new strategy for skeleton pruning. *Pattern Recognition Letters*, 2015. (Cité pages 53 et 55.)

Doron Shaked et Alfred M Bruckstein. The curve axis. *Computer Vision and Image Understanding*, 63(2) :367–379, 1996. (Cité page 54.)

Doron Shaked et Alfred M Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 69(2) :156–169, 1998. (Cité page 54.)

Wei Shen, Xiang Bai, Rong Hu, Hongyuan Wang, et Longin Jan Latecki. Skeleton growing and pruning with bending potential ratio. *Pattern Recognition*, 44(2) :196–209, 2011. (Cité pages xii, 52, 53 et 55.)

Wei Shen, Yan Wang, Xiang Bai, Hongyuan Wang, et Longin Jan Latecki. Shape clustering : Common structure discovery. *Pattern Recognition*, 46(2) :539–550, 2013. (Cité pages 59, 79, 129 et 144.)

Frank Y Shih et Christopher C Pu. A skeletonization algorithm by maxima tracking on euclidean distance transform. *Pattern Recognition*, 28(3) :331–341, 1995. (Cité page 42.)

Ali Shokoufandeh, Lars Bretzner, Diego Macrini, M Fatih Demirci, Clas Jönsson, et Sven Dickinson. The representation and matching of categorical shape. *Computer Vision and Image Understanding*, 103(2) :139–154, 2006. (Cité pages xiii, 57 et 58.)

Ali Shokoufandeh, Diego Macrini, Sven Dickinson, Kaleem Siddiqi, et Steven W Zucker. Indexing hierarchical structures using graph spectra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7) :1125–1140, 2005. (Cité page 57.)

Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, et Steven W Zucker. Hamilton-jacobi skeletons. *International Journal of Computer Vision*, 48(3) :215–231, 2002. (Cité pages viii, xiv, 41, 65, 67, 78, 79, 80, 82, 83, 84, 88, 90, 92 et 93.)

Kaleem Siddiqi, Ali Shokoufandeh, Sven J Dickinson, et Steven W Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1) :13–32, 1999. (Cité pages 47, 57 et 84.)

Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, et Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12) :1349–1380, 2000. (Cité page 2.)

Milan Sonka, Vaclav Hlavac, et Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014. (Cité page 34.)

Juan Humberto Sossa-Azuela et al. On the computation of the euler number of a binary object. *Pattern recognition*, 29(3) :471–476, 1996. (Cité page 35.)

Daniel A. Spielman et Shang-Hua Teng. Smoothed analysis of algorithms : Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3) :385–463, 2004. (Cité page 147.)

Olarik Surinta, Mahir F Karaaba, Lambert RB Schomaker, et Marco A Wiering. Recognition of handwritten characters using local gradient feature descriptors. *Engineering Applications of Artificial Intelligence*, 45 :405–414, 2015. (Cité page 2.)

Gabriel Taubin et David B Cooper. *Object Recognition Based on Moment (or Algebraic) Invariants*. IBM TJ Watson Research Center, 1991. (Cité page 35.)

Cho-Huak Teh et Roland T Chin. On image analysis by the methods of moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4) :496–513, 1988. (Cité page 35.)

Edouard Thiel. Unification de la squelettisation menée en distance. *Reconnaissance des Formes et Intelligence Artificielle*, 1 :349–358, 1992. (Cité page 41.)

Antoine Vacavant et Aurélie Leborgne. Robust computations of reeb graphs in 2-d binary images. Dans *Computational Topology in Image Context*, 2016. (Cité page 162.)

Junwei Wang, Xiang Bai, Xinge You, Wenyu Liu, et Longin Jan Latecki. Shape matching and classification using height functions. *Pattern Recognition Letters*, 33(2) :134–143, 2012. (Cité page 34.)

Rei-Yao Wu et Wen-Hsiang Tsai. A new one-pass parallel thinning algorithm for binary images. *Pattern Recognition Letters*, 13(10) :715–723, 1992. (Cité page 39.)

Yun Xia. Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10) : 1076–1086, 1989. (Cité page 38.)

Jun Xie, Pheng-Ann Heng, et Mubarak Shah. Shape matching and modeling using skeletal context. *Pattern Recognition*, 41(5) :1756–1767, 2008. (Cité pages 34 et 165.)

Yao Xu, Bo Wang, Wenyu Liu, et Xiang Bai. Skeleton graph matching based on critical points using path similarity. Dans *Asian Conference on Computer Vision*, pages 456–465. Springer, 2009. (Cité page 129.)

Cong Yang, Oliver Tiebe, Kimiaki Shirahama, et Marcin Grzegorzek. Object matching with hierarchical skeletons. *Pattern Recognition*, 2016. (Cité pages xii, 33, 51, 52, 53 et 55.)

Mingqiang Yang, Kidiyo Kpalma, et Joseph Ronsin. A survey of shape feature extraction techniques. *Pattern recognition*, pages 43–90, 2008. (Cité page 35.)

Q.-Z. Ye. The signed euclidean distance transform and its applications. *International Conference on Pattern Recognition*, pages 495–499, 1988. (Cité pages xi, 25, 28, 41 et 78.)

Y. Yokoi, J.I. Toriwaki, et T. Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Journal of Computer Graphics and Image Processing*, 4 :63–73, 1975. (Cité page 14.)

Ian T Young, Joseph E Walker, et Jack E Bowie. An analysis technique for biological shape. i. *Information and control*, 25(4) :357–370, 1974. (Cité page 34.)

Dengsheng Zhang et Guojun Lu. Generic fourier descriptor for shape-based image retrieval. Dans *International Conference on Multimedia and Expo*, volume 1, pages 425–428, 2002. (Cité page 35.)

Dengsheng Zhang et Guojun Lu. Review of shape representation and description techniques. *Pattern recognition*, 37(1) :1–19, 2004. (Cité pages 33 et 35.)

Dengsheng Zhang, Guojun Lu, et al. A comparative study of fourier descriptors for shape representation and retrieval. Dans *Proc. 5th Asian Conference on Computer Vision*, 2002. (Cité page 35.)

Song Chun Zhu et Alan L Yuille. Forms : a flexible object recognition and modelling system. *International Journal of Computer Vision*, 20(3) :187–212, 1996. (Cité page 61.)