N° d'ordre : 04 / 2006-M / IN

République Algérienne Démocratique et populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumédiène
Faculté de Génie Electrique

Mémoire

présenté pour l'obtention du diplôme de :

MAGISTER EN INFORMATIQUE

Spécialité : Programmation et système

Mlle Afifa DAHMANE

Sujet :

Contribution au codage textuel d'images

Soutenu le 12/07/2006, devant le jury suivant :

Mr. H. AZZOUNE, Maître de conférences, USTHB	Président
Mr. S. LARABI, Maître de conférences, USTHB	Directeur de thèse
Mme. F. SOUAMI, Maître de conférences, USTHB	Examinatrice
Mr. R. BABA ALI, Maître de conférences, USTHB	Examinateur

Résumé

Dans le domaine de l'échange d'informations, les images constituent une partie importante des données qui transitent sur les réseaux.

Cependant, la taille de ces images est beaucoup plus importante que celle du texte. Un décalage existe entre les possibilités matérielles qui existent et les besoins exprimés par les utilisateurs ainsi que les applications multimédia.

Des techniques de compression, ont alors été élaborées pour diminuer la taille du stockage et le temps de transmission. Ces méthodes couramment utilisées sont efficaces mais pas suffisantes.

Dans ce cadre, nous présentons une nouvelle écriture du langage de description textuelle d'images LWDOS, que nous avons nommé XLWDOS.

Cette écriture est basée sur la représentation de formes et s'écrit en standard XML. XLWDOS permet, en plus de sa petite taille, une facilité de traitement.

En effet, le descripteur XML peut contenir des informations concernant la sémantique de l'image. Ces informations permettent une exploitation efficace dans le domaine de recherche d'image par le contenu.

Afin de concrétiser la faisabilité et l'efficacité du codage / décodage XLWDOS, la visualisation de descripteurs a été réalisée et le résultat étudié.

Mots clés : LWDOS, XLWDOS, compression d'images, codage / décodage d'images, représentation de formes, XML, Java.

Dédicaces

Je dédie mon mémoire à :

- mes familles, branches DAHMANE et RAYANE, pour m’avoir encouragée et soutenue avec constance.
- mes amis proches pour leur critiques et observations utiles.

Remerciements

Je tiens tout d’abord à remercier monsieur S.LARABI pour m’avoir guidée et accompagnée tout au long de l’élaboration de ce travail.

Je remercie également Mme. Souami, Mr Azoune et Mr. Baba Ali pour avoir accepté d’être membre du jury.

Sommaire

Chapitre3 : Ecriture XML d'un descripteur LWDOS

Chapitre 4 : Reconstruction de l'image à partir de son descripteur XLWDOS

Chapitre 5 : Validation expérimentale

Introduction

Les images, éléments incontournables d'une page Web, permettent au visiteur de mieux identifier le contenu du site et, dans certains cas, évitent d'avoir recours à des textes trop explicatifs et trop longs.
Dans la visualisation d'une page Web, les derniers éléments à êtres affichés sont les images parce que leur taille est beaucoup plus importante que celle du texte.
Chaque pixel a une *profondeur* qui représente le nombre de couleurs différentes qu'il peut avoir. Ainsi dans une image de 16 000 000 couleurs, un pixel sera codé sur 24 bits**.**

Le format numérique des images est donc extrêmement coûteux en taille mémoire. Pour résoudre ce problème de coût qui peut limiter la faisabilité de stockage et de transmission des images, des techniques de compression d'images ont été élaborées pour compacter leur représentation numérique.

A l'aide de ces techniques de compression, le stockage et la transmission des images sont plus efficaces et plus rapides.

Mais avant que quiconque ne puisse utiliser une image, il est nécessaire de commencer par la localiser. Au même moment, l'augmentation du nombre d'informations potentiellement intéressantes rend la recherche de plus en plus difficile.
Des bases de données multimédias permettent aujourd'hui de chercher sur le marché des images à partir de certaines caractéristiques comme la couleur, la texture ou la forme d'objet dans l'image.

Dans cet axe de recherche, une méthode de description de silhouette a été développée au laboratoire L.R.I.A [15]. Cette description est caractérisée par un contenu sémantique structuré. Un langage LWDOS (Language for Writing Descriptors of Outline Shapes) a été aussi proposé pour l'écriture textuelle de cette description.
Un descripteur LWDOS occupe peu d'espace mémoire et offre donc de meilleures conditions de stockage et de transfert d'images. La structuration des informations sémantiques qu'il contient facilite son indexation. Ces dernières informations peuvent êtres exploitées dans les applications utilisant le CBIR (recherche d'images basée sur le contenu).

Problème posé :

Le traitement par une application d'un texte brut non formaté est difficile. Ce traitement peut consister en l'indexation, la classification dans une base de données ou simplement le décodage d'un descripteur afin de reconstituer l'image.

Afin de pallier ce problème et rendre la manipulation de descripteurs textuels d'images plus efficace, il est judicieux d'imposer un format standard que n'importe quel descripteur doit respecter.

Le langage de balisage XML (eXtensible Markup Language) offre une syntaxe générique utilisée pour formater les données. Les DTD (Document Type Definition) sont utilisées pour définir la structure du document XML, c'est-à-dire, les noms de ses éléments ainsi que leurs attributs, l'ordre de leur apparition … etc.

La syntaxe de LWDOS peut être adaptée à ce format balisé et être définie par une DTD. Ainsi, nous nous assurons que toutes les informations sur la silhouette sont contenues dans le document.

Il s'agit ensuite de proposer une méthode pour le décodage d'un descripteur LWDOS et sa visualisation. On parlera alors d'un nouveau format d'images dont la nouveauté est le texte utilisé avec une structure XML.

Le travail que nous avons réalisé et que nous présentons dans ce mémoire est scindé en cinq chapitres.

Dans le premier chapitre, nous donnons un aperçu sur la manière dont les images sont compressées, stockées et représentées.

Dans le chapitre suivant, nous présentons avec détails la méthode, de description textuelle de silhouettes, développée au laboratoire L.R.I.A. ainsi que le langage LWDOS pour l'écriture de ces descriptions.

Dans le troisième chapitre, nous proposons une nouvelle écriture du descripteur textuel LWDOS formatée en XML que nous avons nommé XLWDOS.

Dans le quatrième chapitre, nous montrons comment exploiter un descripteur XLWDOS pour le décoder et visualiser l'image décrite.

Au dernier chapitre, la méthode que nous avons développée a été implantée et les résultats obtenus sont présentés et commentés.

Nous terminons par une conclusion dans laquelle nous rappelons les résultats obtenus ainsi que les perspectives à ce travail.

Chapitre 1 :

L’image numérique

1.1. Introduction

Diverses applications utilisent des images sous format numérique. Ces applications sont basées sur la capacité de stocker et de transmettre les images.
Cependant, le format numérique des images est extrêmement coûteux en taille mémoire, ainsi qu'en bande passante des réseaux qui n'augmente pas aussi rapidement que la puissance des machines.

Les applications de traitement d'images exploitent la puissance des machines pour faciliter la manipulation de ces données visuelles très utilisées.
La facilité de manipulation des images peut dépendre de la manière dont ces données sont classées, représentées dans des bases de données et même le format sous lequel elles sont stockées.

Dans ce chapitre, nous présentons d'abord quelques méthodes de compression d'images. Nous donnons ensuite un recueil des méthodes de représentation d'images utilisées pour des problèmes de reconnaissance, de classification ou de recherche des formes.

1.2. La compression d'image

Afin de contrecarrer les problèmes de taille des images et de satisfaire les besoins de stockage et de transmission, il est courant de réduire la taille des données.

Pour cela, des techniques de compression d'images ont été proposées.

1.2.1. Définitions

1.2.1.1. Le codage Luminance, Chrominance

Il est reconnu depuis longtemps que notre oeil est plus sensible à la luminance (l'intensité de la lumière) qu'à la chrominance (les couleurs). La première technique pour réduire la taille d'une image est donc de donner plus d'importance aux informations de luminance que de chrominance qui seront codées sur un espace moins important.
La compression tient dans le fait que l'on supprime carrément des informations de chrominance [11].

1.2.1.2. Compression physique et logique

La compression physique agit directement sur les données, il s'agit ainsi de regarder les données redondantes.

La compression logique par contre est effectuée par un raisonnement logique en substituant une information par une information équivalente [19].

1.2.1.3. Compression symétrique et asymétrique

Dans le cas de la compression symétrique, la même méthode est utilisée pour compresser et décompresser l'information. Il faut donc la même quantité de travail pour chacune de ces opérations.

Par contre, la compression asymétrique demande plus de travail pour l'une des deux opérations [32].

1.2.2. Méthodes de compression d'image

Les techniques de compression se divisent en deux catégories principales : « compression sans perte» et « compression avec perte ».

1.2.2.1. Compression sans perte :

Elle consiste à enlever la redondance dans les données juste nécessaire pour représenter l'image. Cette redondance est directement liée à la prédictibilité des éléments constituants de l'image. Par exemple, une image de couleur unie est totalement redondante du fait que la couleur fournit suffisamment d'information pour représenter toute l'image.

La compression sans perte identifie les éléments constituants de l'image et exploite leur structure pour réduire la quantité des données.

Ces compressions sont applicables à tous les types de données mais les taux de compression reste la plupart du temps faibles.

1.2.2.1.1. Méthode RLE

C'est l'un des principes les plus simples. Au lieu de coder plusieurs fois une valeur lorsqu'elle apparaît plusieurs fois à la suite, on code le nombre de fois ou elle est présente. Ainsi, selon ce principe, la figure ci-dessous représentant un extrait de 16 pixels en noire et blanc peut être compressée en : 5 255 1 000 10 255

255	255	255	255
255	000	255	255
255	255	255	255
255	255	255	255

Sans compression :16 octets.

Compressée :6 octets.

Nous avons un gain de compression de 10/16 %. Cependant, la compression RLE n'a du sens que pour les données possédant de nombreux éléments consécutifs redondants, notamment les images possédant de larges parties uniformes [11].

1.2.2.1.2. Le codage de Huffman

D.A. Huffman a inventé en 1952 un algorithme de compression capable, à partir d'une analyse statistique des données, d'associer à celles le plus souvent présentes, les codes les plus courts. Inversement, les données les plus rares se verront attribuer les codes les plus longs.
Prenons l'exemple suivant :

b	b	b	b
b	b	r	r
n	r	r	r
n	r	r	r

B : 6/16
R : 8/16
N : 2/16

Pour rechercher le code, on utilise l'arbre de Huffman en mariant les deux pourcentages les plus faibles.

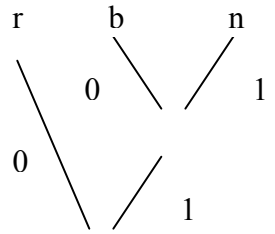

Figure 1.1 : arbre de Huffman

Ceci nous donne :
R : 0
B : 10
N : 11

Cet algorithme permet d'obtenir de bons résultats, mais il faut conserver, entre la compression et la décompression, le dictionnaire des codes utilisés [11].

1.2.2.1.3. La compression LZW :

Du nom de ses auteurs, Lempel, Ziw et Welch, cette méthode vise à déceler des suites de bits ou d'octets similaires, autrement dit à identifier des motifs qui se retrouvent dans la succession de valeurs décrivant l'image. Chaque fois qu'une telle suite est rencontrée, l'algorithme la range dans un dictionnaire et la remplace par un identificateur.

Le dictionnaire est initialisé avec les 256 valeurs de la table ASCII. Le fichier à compresser est découpé en chaînes d'octets (ainsi pour des images monochromes - codées sur 1 bit - cette compression est peu efficace), chacune de ces chaînes est comparée au dictionnaire et est ajoutée si jamais elle n'y est pas présente. L'algorithme parcourt le flot d'informations en le codant; Si jamais une chaîne est plus petite que le plus grand mot du dictionnaire alors elle est transmise [16].

Etant donnée que cette méthode est de type dictionnaire, la compression et la décompression se font instantanément, et n'a pas besoin d'opération intermédiaire.

1.2.2.2. Compression avec perte :

Le but est de perdre le minimum d'information de manière à garder un aspect visuel le plus proche.

1.2.2.2.1. La compression JPEG :

Le principe de l'algorithme JPEG pour une image à niveau de gris est le suivant [11]:

- la matrice des pixels de l'image numérique est décomposée en blocs de 8 x 8 pixels qui vont tous subir le même traitement. Une transformation linéaire, le plus souvent du type FFT (Fast Fourier Transform) ou DCT (Discret Cosinus Transform) est réalisée sur chaque bloc. Ces transformations complexes concentrent l'information sur l'image en haut et à gauche de la matrice.
- les coefficients de la transformée sont ensuite quantifiés à l'aide d'une table de 64 éléments définissant les pas de quantification. Cette table permet de choisir un pas de quantification important pour certaines composantes jugées peu significatives visuellement.
- des méthodes sans perte sont ensuite réalisées en utilisant les propriétés statistiques de l'image.

1.2.2.2.2. La compression JPEG 2000 :

Depuis 1995, les experts cherchent à perfectionner le mode de compression du format JPEG. Une technique de compression par ondelettes (DWT : Discrete Wavelet Transform), réputée être deux fois plus efficace que la méthode DCT, sert de base au nouveau format intitulé JPEG2000 [11].

1.3. Les formats d'images

Le format de données est la manière utilisée pour représenter des données sous forme de nombres binaires.
Un format d'image comprend en général un en-tête qui contient des données sur l'image (taille de l'image en pixels par exemple) suivie des données de l'image. La structuration des données est différente pour chaque format d'image.

1.3.1. Le format GIF (bitmap)

Le format GIF (*Graphic Interchange Format*) est un format de fichier graphique bitmap par la société *Compuserve* [5].
Le format Gif est l'un des deux formats standards du web avec le JPEG.
Les bonnes performances du format GIF proviennent d'une part, de la limitation du nombre de couleurs de l'image à 256 au maximum. Et d'autre part, de l'utilisation d'une technique de compression (LZW) bien adaptée au dessin au trait.

Le format GIF est orienté vers la petite taille en générale en kilo-octets. Il est adéquat pour traiter les images en aplats de couleur et n'est pas recommandé pour les images complexes dans l'impression.
Il existe deux formats: GIF87a, GIF89a :

- Le format GIF87a a été créé en 1987. Son format est standard (image simple)

- Le format GIF89a, qui a été amélioré en 1989, possède une possibilité de transparence de couleur, d'animation d'images, et d'effet d'entrelacement.

L'animation : Le GIF animé est une méthode simple : une succession d'images à un même emplacement. Lorsque la première image est ouverte, la deuxième vient alors se placer par-dessus, et ainsi de suite.… jusqu'à la dernière.
Le GIF animé contient donc plusieurs images avec des informations sur la vitesse de défilement, le nombre de répétition.
La transparence : L'une des couleurs de la palette peut être déclarée transparente, ou seulement une zone possédant cette couleur. Si on déclare transparent le fond d'une image GIF, on donne l'impression que l'image a été découpée en suivant le bord de la partie non transparente [5, 14].
L'entrelacement : L'image GIF peut s'afficher de manière progressive (image gif entrelacée). Elle est alors partagée en quatre sous-images, qui sont transmises et affichées l'une après l'autre. Si la transmission est lente, et / ou si l'image est lourde, l'internaute peut décider, au vu de ce qui est déjà affiché, de patienter ou non pour avoir l'image finale.

1.3.2. Le format BMP

Le format BMP est un des formats les plus simples développé conjointement par Microsoft et IBM, Un fichier BMP est un fichier bitmap. Il est caractérisé par son adaptabilité à une image complexe dont les formes ne suivent pas un ordre logique mais le poids de l'image est conséquent et nécessite une grosse capacité de traitements.
La structure d'un fichier bitmap est la suivante [14]:

- En-tête du fichier (*file header*) : fournit des informations sur le type de fichier (Bitmap), sa taille et indique où commencent les informations concernant l'image à proprement parler.

- En-tête du bitmap (*bitmap information header*) : fournit des informations sur l'image, notamment ses dimensions et ses couleurs.

- Palette (optionnellement) : La *palette* n'est présente que pour les images d'au plus 256 couleurs, soit 8 bit par pixels. Lorsqu'une palette est définie, elle contient successivement 4 octets pour chacune de ses entrées représentant la composante bleue (sur un octet), verte (sur un octet), rouge (sur un octet) et un champ réservé (sur un octet).

- Corps de l'image : Le codage de l'image se fait en écrivant successivement les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche.

1.3.3. Le format PNG

Le format PNG (Portable Network Graphics) fait partie du bitmap, il vise à remplacer le format GIF de Unisys qui n'est pas libre de droit. Le GIF payant utilise la compression LZW. Or PNG utilise une autre compression dérivée du LZ77 qui est la base du LZW. L'image en PNG est 10 % à 30 % plus légère que celles en GIF. En plus, il est plus intéressant que GIF pour la transparence d'image car ce dernier n'a qu'une seule couleur transparente. Par contre le format *png* supporte plusieurs niveaux de transparence de 1 à 256, de complètement opaque à complètement transparent [19, 31].
Le PNG peut contenir de l'information de type Méta fichier, information non affichable qui identifie le contenu de l'image, l'auteur, etc.

1.3.4. Le format SWF – Flash

Aujourd'hui, Flash en est à sa sixième version. Il est utilisé par de nombreux développeurs de sites web, et plus de 90 % des internautes sont équipés du plug-in correspondant. Cependant ce format est la plupart du temps utilisé pour réaliser des animations, et n'est presque pas utilisé pour réaliser des images vectorielles fixes [14].

1.4. La représentation de formes

La représentation de formes est utilisée pour des problèmes de reconnaissance, de classification ou de recherche des formes.

La reconnaissance des formes trouve des applications dans une variété de domaines dont notamment l'analyse de documents, les arts visuels, l'Internet, la médecine, la biologie, la physique, l'industrie et la sécurité.

Aussi, la recherche de formes, connaît une forte demande. Des bases de données multimédias permettent aujourd'hui de chercher sur le marché des images à partir de certaines caractéristiques comme la couleur, la texture ou la forme d'objet dans l'image.

1.4.1. Le CBIR

La recherche d'images basée sur le contenu (CBIR) décrit le processus de recherche d'images désirées dans une grande collection sur la base de critères (tels que la couleur, la texture et la forme) qui peuvent être automatiquement extraits à partir des images elles-mêmes. Les critères utilisés pour la recherche peuvent être primitifs ou sémantiques, mais le processus d'extraction est principalement automatique.

Une des questions clés est la nécessité d'extraire l'information utile à partir des données brutes avant que n'importe quel genre de raisonnement au sujet du contenu de l'image soit possible. Les bases de données d'image diffèrent ainsi fondamentalement des bases de données des textes [24].

1.4.2. Classification des techniques de représentation de forme

Plusieurs méthodes de classification sont disponibles.

1.4.2.1. Méthodes basées sur le contour ou sur les régions

C'est la classification la plus commune et la plus générale, elle est basée sur l'utilisation des points de frontière ou bien points d'intérieur d'une forme [22].

Techniques de représentation de formes basées sur le contour :

Les techniques de représentation de forme par le contour exploitent seulement l'information de la frontière de la forme. Il y a généralement deux types d'approches très différentes pour modeler ceci: approche continue (globale) et approche discrète (structurale). Les approches continues ne divisent pas la forme en parties secondaire, un vecteur de caractéristiques dérivé de la frontière est employé pour décrire la forme. Les approches discrètes divisent la frontière de la forme en segments, appelés les primitifs en utilisant un critère particulier. La représentation finale est habituellement un graphe (ou un arbre), la mesure de similitude est faite par l'appariement des graphes.

Techniques de représentation de formes basées sur les régions :

Dans les méthodes basées sur la région, tous les pixels d'une région sont calculés pour obtenir la représentation de la forme. Ces méthodes utilisent les moments, la grille, la matrice de forme, l'enveloppe convexe et l'axe de médian pour décrire la forme.

1.4.2.2. Méthodes IP et NIP

Une autre classification des techniques de représentation de forme est basée sur la conservation de l'information. Les méthodes qui tiennent compte de la reconstruction précise d'une forme à partir de son descripteur s'appellent *information préserving* (IP), alors que les méthodes seulement capables de la reconstruction partielle ou de la description ambiguë s'appellent *non information préserving* (NIP). Pour les systèmes CBIR, l'IP n'est pas une condition [6].

1.4.2.3. Méthodes 2D et 2D/3D

Il est également possible de partager les techniques de représentation de forme en deux familles, les approches 2D et les approches 2D/3D [23].

Le principe de l'approche 2D/3D consiste à associer aux objets 3D un ensemble de projection 2D, correspondant à différents angles de vue. La forme 3D est alors indirectement représentée par divers DF (Descripteurs de Forme) 2D associés à ces images de projection. Notons en outre que ce principe permet également de réaliser des appariements entre modèles 3D et objets 2D extraits des bases d'images ou de vidéos.

Ce principe est actuellement pris en compte dans le standard MPEG-7, dans le cadre du schéma de description *MultiView DS*, qui définit un "conteneur" permettant d'intégrer l'ensemble des descripteurs 2D, qu'ils soient de forme, de couleur ou encore de texture. En ce qui concerne la mesure de similarité, elle est dérivée à partir des mesures de similarité associées aux descripteurs 2D.

1.4.2.4. Méthodes globales et structurales

Techniques de représentation de forme Globales :

Les méthodes globales peuvent êtres statistique, par transformée ou variationnelle.

Les approches par transformée sont caractéristiques du domaine de la reconnaissance. Les représentations d'objets 3D fondées sur des transformées visent à déterminer des représentations de forme globales, définies en terme de transformation intégrale.

Les approches variationnelles s'appuient sur une modélisation physique des déformations que peut subir une surface 3D donnée. Elles recherchent la solution de l'équation d'équilibre dynamique entre tensions internes et champ de forces externes.

Les DFs statistiques consistent en général soit à calculer divers moments statistiques, [13], soit à estimer la distribution de la mesure d'une primitive géométrique donnée (points, cordes, sécantes, triangles, tétraèdres). Utiliser directement les moments statistiques pour la reconnaissance nécessite une normalisation en taille et en position de l'objet, afin d'obtenir une certaine invariance géométrique extrinsèque de la représentation.

Techniques de représentation de forme Structurales :

Contrairement aux approches statistiques, les représentations structurales visent à décrire la notion de forme de manière plus complète et plus intuitive. Un premier type d'approche s'appuie sur une segmentation initiale de l'objet en sous parties satisfaisant certains critères d'homogénéité par rapport à un attribut de forme préétabli, et représentée par des structures spécifiques comme des arbres ou des graphes. Si l'étape de segmentation vise à identifier les différentes structures élémentaires qui composent l'objet considéré, la deuxième phase permet de représenter les relations d'adjacence et éventuellement les positions relatives de ces différentes structures.

1.4.3. Quelques exemples de méthodes de représentation de forme

Nous avons recensé quelques méthodes de représentation de forme**,** que nous présentons en deux groupes : méthodes basées sur le contour et méthodes basé sur les régions.

1.4.3.1. Méthodes basées sur le contour

Curvature Scale Space :
Le descripteur CSS a été proposé initialement par Mokhtarian [10] et retenu pour la norme MPEG7. Son principe de base repose sur la description des concavités d'une courbe et son filtrage successif afin d'y suivre l'évolution de ces concavités. Le rôle du filtrage est de lisser la courbe et donc de faire disparaître progressivement ses concavités.

Descripteurs Simples de Forme :
Les descripteurs globaux simples communs sont la surface, la circularité ($perimeter^2$/surface), l'excentricité (longueur de l'axe principal / longueur de l'axe mineur), l'orientation principale d'axe, et l'énergie de recourbement [13]. Ces descripteurs globaux simples habituellement peuvent seulement distinguer des formes avec de grandes dissimilitudes, donc, ils sont habituellement employés comme filtres ou combinés avec d'autres descripteurs de forme pour distinguer des formes.

Signature de Forme :
Les signatures de forme représentent la forme par une fonction dimensionnelle dérivée des points de frontière de forme. Beaucoup de signatures de forme existent, tels que les coordonnées complexes, les coordonnées polaires, la distance centrale, l'angle de tangente, l'angle cumulatif, la courbure et la surface [7].

Moments de Frontière :
Les moments de frontière peuvent être employés pour réduire la dimension de la représentation par frontière [17].
Transformée Spectrale:
Les descripteurs spectraux surmontent les problèmes de sensibilité au bruit et de variations de frontière en analysant la forme dans le domaine spectral. Les descripteurs spectraux incluent les descripteurs de Fourier (FD) [12].

Décomposition en polygone :
Dans [26] et [27], la frontière de forme est décomposée en segments par approximation polygonale. Les sommets du polygone sont employés comme primitifs. Chaque primitif est exprimé comme liste de quatre éléments qui se compose de l'angle interne, de la distance du prochain sommet, et de ses coordonnées x et y.

1.4.3.2. Méthodes basées sur les régions

Descripteur ART (*Angular Radial Transform*) :
ART est le descripteur basé sur les régions retenu par MPEG7. La forme initiale doit être préalablement mise à l'échelle. Cela est réalisé en centrant l'objet à l'origine du repère et en normalisant sa taille à la distance maximale entre ses points et l'origine du repère. Cela assure une certaine invariance extrinsèque aux translations et homothéties. Quant à l'invariance aux rotations, elle peut être obtenue de manière intrinsèque, en considérant uniquement les valeurs absolues des coefficients de la décomposition [25].
Le descripteur ART présente l'avantage de la généralité, étant applicable à toute forme.

Invariants géométriques de moment :
En utilisant des combinaisons non linéaires des moments d'ordre inférieurs, un ensemble d'invariants de moments (habituellement appelés moments géométriques), qui a les propriétés souhaitables d'être invariable sous la translation, le changement d'échelle et la rotation, est dérivé. Les invariants géométriques de moment ont attiré une attention large [18] et ont été employés dans des applications.

Invariants algébriques de moment :
Les invariants algébriques de moment sont calculés à partir du premier centre *m* de moment et sont donnés comme valeurs propres des matrices prédéfinies, *M[j, k]*, dont les éléments sont des facteurs mesurés des moments centraux.
Différent des invariants géométriques de moment de Hu, les invariants algébriques de moment peuvent être construits jusqu'à un ordre arbitraire [3].

Méthode basée sur les grilles :
Fondamentalement, une grille d'un certain nombre de cellules est étendue sur une forme, la grille est alors balayée de gauche à droite et du haut vers le bas. Le résultat est une carte binaire. Les cellules couvertes par la forme sont assignées à 1 et les autres à 0. La forme est représentée comme vecteur binaire. La *binary Hamming distance* ou la *city block distance* est employée pour mesurer la similitude entre deux formes [13].

La matrice de forme :
L'utilisation de la matrice de forme qui est dérivée d'une technique d'échantillonnage circulaire de trame [26]. Plutôt que de placer la grille carrée sur une image de forme, une trame polaire de cercles concentriques et de lignes radiales est étendue au centre de l'image La valeur binaire de la forme est prélevée aux intersections des cercles et des lignes radiales. La matrice de forme est formée de sorte que les cercles correspondent aux colonnes de la matrice et les lignes radiales correspondent aux rangées de la matrice.

Enveloppe Convexe (Convex Hull) :
Une région R est convexe si et seulement si pour deux points quelconques x1, x2 ∈ R, le segment x1x2 est à l'intérieur de la région. L'enveloppe convexe d'une région est la plus petite région convexe H qui satisfait la condition R ⊂ H. La différence R - H s'appelle l'insuffisance convexe D de la région R. Extraire l'enveloppe convexe peut être un processus simple qui trouve des insuffisances convexes significatives le long de la frontière. La forme peut alors être représentée par une chaîne de concavités. Une représentation complète de la forme peut être obtenue par un processus récursif qui a comme conséquence un arbre de concavités.

Axe Médian :
Comme l'enveloppe convexe, le *squelette* de région peut également être utilisé pour la représentation et la description de forme. Un *squelette* peut être défini comme ensemble de lignes médianes des parties d'une forme [12].
L'axe médian est le lieu des centres des disques maximaux inclus dans la forme. Le squelette de la forme est enveloppé par un rectangle. Le squelette peut alors être décomposé en segments et être représenté comme graphique selon certains critères.

1.5. Conclusion

Les méthodes de compression présentées dans ce chapitre permettent de réduire énormément la taille des images pour leur stockage et transfert. Les méthodes de représentation permettent en plus de la réduction d'espace mémoire, une indexation facile de leurs contenus.

Dans le chapitre suivant, nous présentons la méthode développée au laboratoire LRIA de l'USTHB permettant de produire une description textuelle d'une silhouette. Cette méthode est basée sur la description d'image en la considérant comme étant un ensemble de silhouettes (régions).

Chapitre 2 :

Méthode de description textuelle de silhouettes

2.1. Introduction

La représentation de la forme demeure l'un des problèmes difficiles de la vision par ordinateur. Beaucoup de recherches sur la représentation de formes ont été faites en utilisant les frontières des images de silhouette représentées par une courbe simple fermée [20, 6].
Dans ce chapitre, nous présentons la méthode de description de silhouettes développée au LRIA, ainsi que le langage textuel LWDOS permettant d'écrire textuellement les descripteurs [21].

Le but de cette méthode est de proposer une décomposition sémantique de la silhouette en éléments qui peuvent être écrits avec un langage textuel. La méthode proposée permet de reconstituer entièrement la silhouette à partir de sa description et satisfait un ensemble de critères [10] : Invariance, unicité, invariance au changement d'échelle, à la rotation et à la translation. Il vérifie également les propriétés additionnelles comme la facilité de calcul et de stockage.

2.2. Décomposition des silhouettes en parties

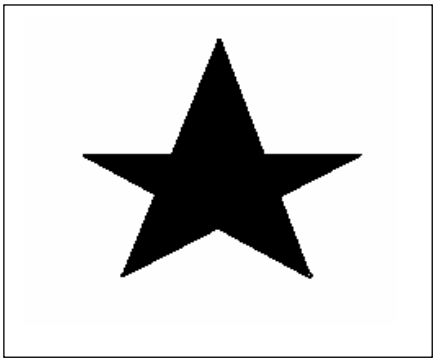

Figure 2.1 : Exemple de silhouette

Nous supposons que la silhouette a été extraite à partir d'une image (voir la figure 2.1). Lors du balayage d'une silhouette suivant une direction choisie, des points de concavité peuvent être rencontrés sur sa frontière (voir la figure 2.2). Aux points concaves où l'orientation du contour change, une décomposition de la silhouette est effectuée. La ligne passant par le point de décomposition est la ligne de partitionnement. Cette ligne décompose la silhouette en partitions.

Afin d'obtenir une décomposition invariante à la rotation de la silhouette, la direction du balayage doit correspondre à celle du rectangle minimum l'englobant.

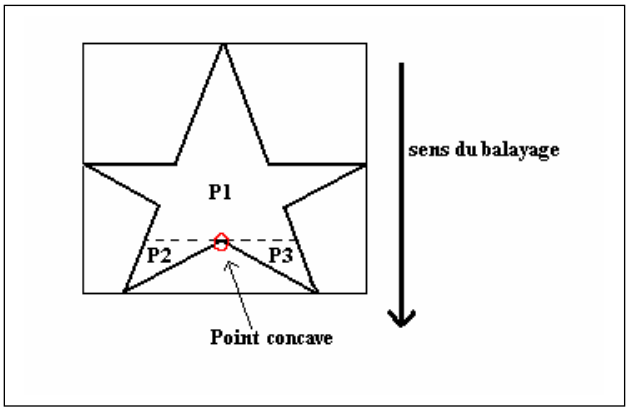

Figure 2.2 Processus de décomposition.

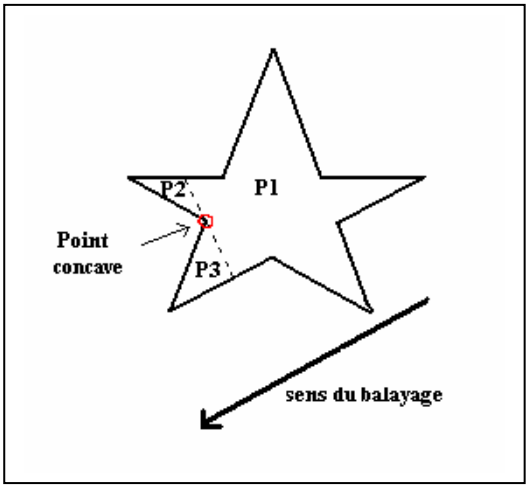

Figure 2.3 Résultat de la décomposition suivant une autre direction.

Pour assurer l'unicité de cette représentation, la description de tous les éléments doit être ajoutée. Dans ce qui suit nous montrons comment les parties peuvent être décrites.

2.3. Description des parties

Chaque partie est définie comme ensemble de deux frontières (gauches et droites) qui commencent au point gauche le plus élevé et se terminent au plus bas point de la silhouette (voir la figure 2.4).
Quand une partie est jointe à d'autres, les extrémités suivantes sont considérées :

- pour les parties venant après la ligne de jonction ou de disjonction, il y a deux points de début localisés comme premiers points des frontières
- pour les parties précédant la ligne de jonction ou de disjonction, il y a deux points de fin situés en tant que derniers points des frontières.

Pour décrire les partitions, les frontières gauches et droites sont segmentées en simples éléments (ligne ou courbe) au moyen des points de courbure. Chaque élément est décrit en utilisant:

- son type qui peut être courbe ou ligne, convexe ou concave. Pour la courbe, le degré de concavité et de convexité sont considérés
- son orientation, donnée comme angle d'inclinaison relativement avec la direction de l'axe Ox
- sa longueur.

La figure 2.5 montre un exemple de description de contour.

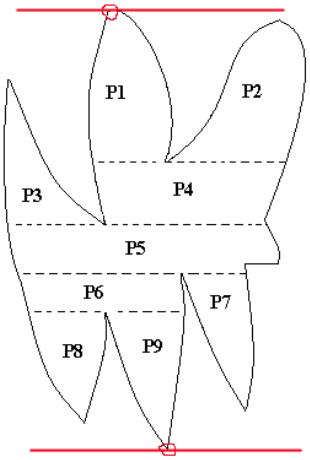

Figure 2.4 : Points de début et de fin

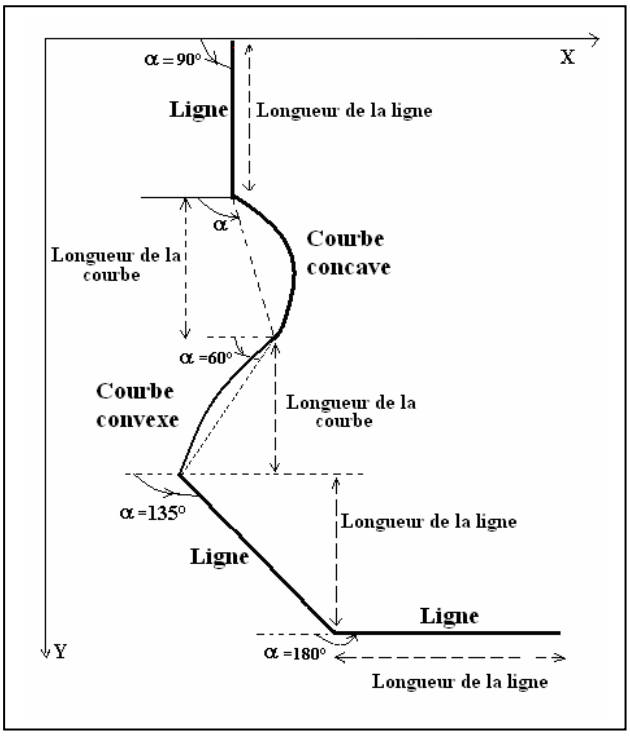

Figure 2.5 : Eléments de description d'une frontière.

2.4. Description des lignes de jonction et de disjonction

Afin de positionner toutes les parties, chaque ligne de jonction et de disjonction est considérée comme ensemble de segments caractérisés par :

- leur type qui peut être **Junction** si elle est commune à deux parties, **Free High** si elle appartient seulement à la partie du haut, **Free Low** si elle appartient seulement à la partie inférieure,
- leur longueur,
- les parties à la quelle elles appartiennent, par exemple la ligne de jonction de la figure 2.6 se compose de cinq segments respectivement de type FreeLow, jonction, FreeHigh, jonction, FreeLow.

Cette description permet de récupérer entièrement la forme géométrique de la partie relativement à la forme d'ensemble.

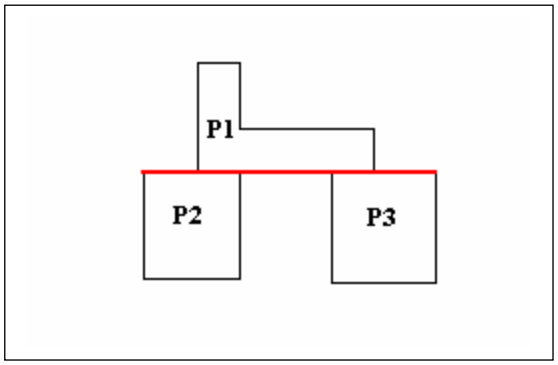

Figure 2.6. Segments de la ligne de jonction

2.5. Syntaxe du langage LWDOS

2.5.1. Description des frontières de la silhouette

Comme il a été présenté ci-dessus, les contours élémentaires sont décrits avec un ensemble d'attributs: type (ligne, courbe, convexe, degré de convexité, concave, degré de concavité), angle d'inclinaison et longueur relative. Les notations **r**, **cv**, **cc** sont employées pour indiquer respectivement le type ligne, convexe, et concave. Ainsi, **r 120 1/5** est la description de la ligne inclinée avec 120° et avec la longueur relative de 1/5. Des frontières gauches et droites sont décrites en utilisant les descriptions de leurs contours élémentaires écrites entre les symboles de délimiteur. Par exemple, la frontière gauche de la partie 1 de la silhouette illustrée par la figure 2.6 est écrite comme ceci :

Left boundary of part 1 → { r 90 1/2 }
Right boundary of part 1 → {r 180 1/6 r 90 1/3 r 180 1/2 r 90 1/6 }

Les partitions sont décrites en utilisant la description de ses deux frontières comme ceci :

{Description de frontière gauche}{Description de la frontière droite}

2.5.2. Description des lignes de jonction et de disjonction

Les segments de la ligne de jonction sont décrits en utilisant les symboles **j, w**, **h** dénotant respectivement les attributs **jonction**, **FreeLow** et **FreeHigh**. Par exemple, "j 1 4 1/10" et "j 2 4 2/10" sont la description des deux segments de la première ligne de jonction de la silhouette de la figure 2.4. La description de la ligne de jonction et de disjonction est écrite entre les délimiteurs comme ceci :

Ligne de jonction: ↑ Description of segment 1 Description of segment 2 …. … ↓
Ligne de disjonction:↓ Description of segment 1 Description of segment 2 …. … ↑

2.5.3. Description des parties composées

Un ensemble de parties (trois au moins) jointes en utilisant la ligne de jonction ou de disjonction constituent une partie composée.
Pour écrire cette composition, la syntaxe suivante est employée :

[Part 1 Part2 …. Part n-1 JL Part n]

Ou bien :
[Part 1 DJL Part2 …. Part n-1 Part n]

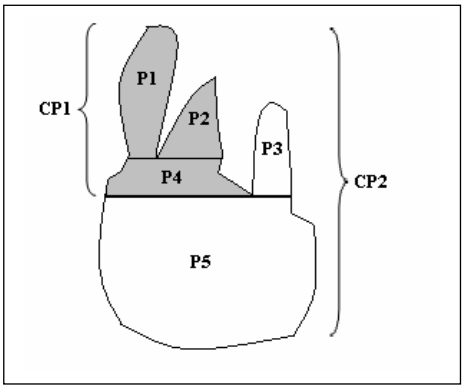

Figure 2.7: partie composée

Une partie composée, considérée comme partie, peut être elle-même être jointe à d'autres parties pour constituer une autre partie composée (voir le schéma 2.7). La première partie composée de la silhouette du schéma 2.7 est écrite comme :
$CP1 = (P_1 , P_2 , JL_1 , P_4)$

La partie composée notée CP1 est jointe avec les parties P3, P5 pour constituer une nouvelle partie composée écrite comme :
$CP_2 = (CP_1 , P_3 , JL_2 , P_5)$

2.5.4. La grammaire LWDOS

Les non terminaux du vocabulaire LWDOS sont donnés dans l'ensemble :
V_N={Outline-Shape-Description, Part, List-Parts, Composed-Part, Left-Boundary, Right-Boundary, Contour-Descriptor, Type-Contour, Angle-Inclination, Junction-Line, Disjunction-Line, Attribute, Relative-Length, Number-High-Part, Number-Low-Part}, où :

- **Outline-Shape-Description** est l'axiome initial.
- **Left-Boundary**, **Right-Boundary** dénotent respectivement les frontières gauche et droite.
- **Contour-descriptor** dénote la description du contour de la frontière gauche ou droite.
- **Type-Contour** dénote le type géométrique du contour.
- **Part** dénote la partie.
- **Junction-Line** et **Disjunction-Line** dénote la ligne qui joint quelques parties.
- **List-Parts** dénote un ensemble de parties qui se joignent avec une autre partie.
- **Composed-Part** dénote la composition de **List-Parts**, **Junction-Line** ou **Disjonction-Line** et une **Part**.
- **Attribute** dénote le type du segment de jonction et disjonction lines et la partie à qui il appartient.
- **Relative-Length** dénote la taille relative
- **Number-High-Part** et **Number-Low-Part** dénote le nombre des parties high et low.

Le vocabulaire terminal de LWDOS est donné par :
V_T ={**j, h, w, cv, cc, r** }
Utilisant les symboles suivants:

- [ ] pour le début et la fin d'une partie composée,
- { } pour le début et la fin de la frontière gauche,
- { } pour le début et la fin de la frontière droite,
- ↑ ↓ pour le début et la fin de la ligne de jonction,
- ↓ ↑ pour le début et la fin de la ligne de disjonction,

Les règles de la grammaire LWDOS sont données par la Table suivante :

¡Outline-Shape-Description¿ ::= ¡Composed-Part¿
¡Composed-Part¿ ::= **[** ¡List-Parts¿ ↑ ¡Junction-Line¿ ↓¡Part¿ **]** /
[¡Part¿ ¿↓ ¡Disjunction-Line¿ ↑ ¡List-Parts¿ **]** /
[¡Composed-Part¿ ↑ ¡Junction-Line¿ ↓ ¡Part¿ **]** /
[¡Part¿ ↓ ¡Disjunction-Line¿ ↑ ¡Composed-Part¿ **]** /
[¡Part¿ **]**
¡Part¿ ::= **{** ¡Left-Boundary¿ **}** **{** ¡Right-Boundary¿ **}** /
{ ¡Left-Boundary¿ **}** **{ s }** /
{ s ¡Number-of-Symmetrical-Part¿ **}** /
{ d ¡Number-of-Identical-Part¿ **}** / ε
¡List-Parts¿ ::= ¡Part¿ ¡List-Parts¿ / ¡Part¿ ¡Composed-Part¿ / ¡Composed-Part¿ ¡List-Parts¿ / ¡Part¿ / ε
¡Left-Boundary¿ ::= ¡Contour-Descriptor¿ ¡Left-Boundary¿ / ¡Contour-Descriptor¿
¡Right-Boundary¿ ::= ¡Contour-Descriptor¿ ¡Right-Boundary¿ / ¡Contour-Descriptor¿
¡Contour-Descriptor¿::= ¡Type-Contour¿ ¡Angle-Inclination¿ ¡Relative-Length¿
¡Type-Contour¿::= **cv** Convexity-degree / **cc** Concavity-degree / **r**

¡Angle-Inclination¿::= REAL-VALUE
¡Relative-Length¿::= REAL-VALUE
¡Junction-Line¿ ::= ¡Attribute¿ ¡Relative-Length¿ / ¡Attribute¿ ¡Relative-Length¿ ¡Junction-Line¿
¡Disjunction-Line¿ ::= ¡Attribute¿ ¡Relative-Length¿ / ¡Attribute¿ ¡Relative-Length¿ ¡Disjunction-Line¿
¡Attribute¿::= **j** ¡Number-High-Part¿ ¡Number-Low-Part¿ /
h ¡Number-High-Part¿ / **w** ¡Number-Low-Part¿
¡Number-High-Part¿ ::= INTEGER-VALUE
¡Number-Low-Part¿ ::= INTEGER-VALUE
¡Number-of-Symmetrical-Part¿::= INTEGER-VALUE
¡Convexity-degree¿ ::= REAL-VALUE
¡Concavity-degree¿ ::= REAL-VALUE

Les symboles entourés par des parenthèses, par exemple ¡Part¿sont des non-terminaux. Les symboles gras, par exemple, **j** sont des terminaux. D'autres symboles, comme INTEGER-VALUE pour des classes des terminaux définis dans un dictionnaire séparé.
Le descripteur LWDOS de la silhouette de la figure 2.6 est écrit comme suit :

Outline-Shape-Description → [P_1 ↓ DJL_1 ↑ P_2 P_3], où :

[**{**r 90 1/2 **}** {r 180 1/6 r 90 1/3 r 180 1/2 r 90 1/6 } ↓w 2 1/6 j 1 2 1/6 h 1 1/3 j 1 3 1/6 h 1 1/6 ↑ **{**r 90 1/2 r 180 1/2**}** **{** r 90 1/2 **}** **{{**r 90 1/2 r 180 1/2**}** **{** r 90 1/2 **}** **]**

2.6. Conclusion

Dans ce chapitre, nous avons présenté la méthode de description de silhouette développée au LRIA, ainsi que le langage textuel LWDOS permettant d'écrire textuellement les descripteurs.
La description obtenue est invariable au changement de l'échelle et à la rotation. Le descripteur LWDOS est textuel, ce qui le rend de taille petite relativement aux formats d'images standard qui existent.

Dans le chapitre suivant, nous présentons une nouvelle écriture du descripteur textuel LWDOS formatée en XML que nous avons nommé XLWDOS.

Chapitre 3 :

Ecriture XML d’un descripteur LWDOS

3.1. Introduction

XML offre une flexibilité d'expression quasi infinie, puisqu'il définit une syntaxe générique utilisée pour formater des données avec des balises simples et compréhensibles par l'homme.

Les applications nécessitant de lire des documents XML spécifiques ne peuvent se permettre autant de flexibilité car pour que le descripteur LWDOS puisse être visualisé, il doit respecter une syntaxe formelle.
La solution à ces problèmes est la DTD. Les DTD définissent avec précision quelles contraintes sont associées à un document XML pour que ce dernier soit conforme à sa DTD.

3.2. Eléments d'un langage XML

3.2.1. Structure d'un document XML

Un document XML peut être :

Un document bien formé : C'est un document qui obéit à la syntaxe du langage XML, il sera alors déclaré *correct* par un parseur XML.

Un document valide : C'est un document *bien formé* qui obéit à certaines contraintes de structuration. La structure ou le format d'un document peut être défini avec les DTD (Document Type Definition) ou bien avec le nouveau langage XML Schéma (ou XSD).

Tout document XML se compose des éléments suivants [8,29] :

3.2.1.1. Le prologue

Un prologue est facultatif dans un document non valide mais conseillé. Il peut contenir la déclaration XML, les instructions de traitement et la déclaration de type de document.

3.2.1.2. L'arbre d'éléments

Un Document XML est formé d'une *hiérarchie* d'éléments formant ainsi un *arbre*, chaque élément comporte une balise d'ouverture, un contenu d 'élément et une balise de clôture sauf pour l'élément vide. Il existe un seul élément racine : *le père.* Chaque élément fils est complètement inclus dans son père : il ne peut pas y avoir de recouvrement d'éléments.
Un élément doit être représenté par la forme :

<nom-elt nom-attr$_1$='val$_1$' nom-attr$_2$='val$_2$'…nom-attr$_n$='val$_n$'>
contenu de l'élément
</nom_elt>.

L'élément vide a la forme suivante :
<nom-elt nom-attr$_1$='val$_1$' nom-attr$_2$='val$_2$' …nom-attr$_n$='val$_n$'/>.
Comme il peut être formé d'une balise ouvrante suivi immédiatement de la balise de fermeture: <nom-elt> </ nom-elt> .

Les noms d'éléments et attributs sont des unités lexicales qui peuvent être formés de caractères alphanumériques et des caractères « - _ . ». L'utilisation du caractère« : » est possible mais il a un sens bien particulier : préfixer des domaines nominaux, qui sera décrit plus loin. Ces noms d'éléments et attributs ne doivent pas contenir d'espaces et ils ne doivent pas commencer par un chiffre, ou par les caractères *« - .»* , ou par la chaîne *xml* qu'elle soit en majuscule, en minuscule ou un mélange des deux.

La chaîne *xml* en majuscule ou minuscule ou une combinaison des deux, apparaissant en début de nom d'attribut est réservée à des usages normalisés définis par le W3C.
Il est à noter que l'ordre des spécifications des attributs d'un élément n'est pas significatif. Aucun nom d'attribut ne peut apparaître plus d'un fois dans la même balise.

Remarque :

Dans les noms d'élément et attributs, les majuscules et les minuscules ne sont pas équivalentes. La différence Majuscules/Minuscules reste une règle générale en XML.
Les noms d'attributs et éléments sont choisis librement par l'utilisateur, par contre quelques noms d'attributs sont réservés [1].

3.2.1.3. Les commentaires

Les commentaires peuvent apparaître dans le contenu d'un élément et dans la déclaration de type du document, toutefois ils ne font pas partie des données textuelles du document et ne sont jamais contenus dans une balise. Un commentaire a la forme suivante :
< !-- Ceci est un commentaire -- >.

Le corps d'un commentaire peut contenir n'importe quel caractère à l'exception de la chaîne «--» et ne se termine pas par un trait d'union «-» . De ce fait un commentaire ne peut pas en contenir un autre et ne se termine pas par « --- >» [1, 29].

3.2.2. Eléments et attributs d'un document XML

3.2.2.1. Balises d'ouverture et de fermeture

Le début de chaque élément XML non vide est marqué d'une balise d'ouverture (balise de début), et sa fin d'une balise de fermeture (balise de fin). La balise d'ouverture commence par « < » et se termine par « > ». Elle contient le nom de l'élément et éventuellement des attributs et leurs valeurs pour décrire certaines propriétés de cet élément. Tandis que la balise de fermeture commence par «</ »et se termine par « > » et contient le nom de l'élément (le même que dans la balise ouvrante). Ce nom spécifie le type de l'élément.

Dans la balise d'ouverture d'un élément, il n'y a pas d'espace entre «< » et le nom de l'élément. Ce dernier, dans la balise de fermeture, peut être suivi d'un ou de plusieurs ou d'aucun espaces.
Chaque attribut est une paire «nom = 'valeur'» et doit être précédé d'un ou plusieurs espaces. La valeur d'un attribut ne peut pas inclure les caractères « ^ % & ». Elle doit être encadrée par des guillemets «" » ou « ' » mais jamais par un mélange des deux. Si la valeur de l'attribut est encadrée par « " » elle peut contenir «'» et vis versa [1, 29].

3.2.2.2. La bonne utilisation des attributs

Il est préférable de faire figurer le contenu informationnel principal d'un document dans le contenu des éléments et de n'utiliser les attributs que pour des paramètres de traitement utilisés par les applications. En utilisant des éléments vides avec des attributs, les informations ne seront pas visualisées (nous pouvons les visualiser avec une feuille de style XSL), cela peut être intéressant dans le cas où on veut *cacher* des informations [1].

3.2.2.3. Contenu d'un élément

Un élément, s'il n'est pas vide, peut contenir des données textuelles, des commentaires, des références à des entités, des sections littérales et des instructions de traitement. Il peut contenir aussi d'autres éléments ou même avoir un contenu récursif.
L'indexation des documents par les moteurs de recherche les plus répandus se fait sur le contenu des élément, voire sur le contenu de certains éléments bien particuliers et pas sur les valeurs d'attributs.

3.2.3. Les DTD

Une DTD (Document Type Definition) est une structure-type prédéfinie. Elle représente l'ensemble de toutes les déclarations contenues, directement ou par référence à des entités externes, dans une déclaration de type de document DOCTYPE. Les déclarations locales au document forment la partie interne de la DTD, et celles contenues dans l'entité externe forment sa partie externe. Si les deux sous-ensembles, interne et externe, sont utilisés, le sous-ensemble interne est considéré comme se produisant avant le sous-ensemble externe. Ceci a pour effet que les déclarations d'entités et de liste d'attributs du sous-ensemble interne ont priorité sur celles du sous-ensemble externe [1].
La DTD contient la déclaration des éléments et attributs du document XML.
Une DTD peut contenir des déclarations d'entités générales, des commentaires mais aussi [1]:

3.2.3.1. Déclaration d'élément

Sa forme est : < !ELEMENT nom-type liste >

liste peut être :

- Elément fils :
  - (nom-type-élt-fils$_1$, nom-type-élt-fils$_2$, ...). En utilisant le connecteur de séquence « , » qui permet de déclarer une suite d'éléments *ordonnés* (l'ordre est imposé).
  - (nom-type-élt-fils$_1$ | nom-type-élt-fils$_2$ | ...). En utilisant le connecteur de choix « | »qui indique qu'un seul élément fils doit être présent dans le document instance.
- Données : en utilisant le mot clef *#PCDATA* (que du texte).
- Modèle mixte : mélange entre données et éléments fils.
- Contenu libre : *ANY*
- Elément vide : *EMPTY*

En plus de ces types d'éléments et des deux connecteurs : «, » et « | », des caractères spéciaux (*, +, ?) permettent de spécifier les contraintes d'occurrences des éléments dans les documents instances :
- Le caractère « ? » : il suit un élément ou un groupe d'éléments et indique qu'il peut y avoir 0 ou 1 occurrence, il indique le caractère optionnel de l'élément.
- Le caractère « * » : il suit un élément ou un groupe d 'éléments et indique qu'il y a 0 ou plusieurs occurrences.
- Le caractère « + » : il suit un élément ou un groupe d'éléments et indique qu'il doit y avoir au moins une occurrence [1,9].
Si aucun de ces caractères spéciaux n'apparaît, l'élément ou le groupe d'éléments apparaît une et une seule fois.

3.2.3.2. Déclaration de liste d'attributs

Sa forme est : < !ATTLIST nom-type-élt nom-attr$_1$ type-attr$_1$ occurrence$_1$… nom-attr$_n$ Type-attr$_n$ occurrence$_n$>.

type-attr$_i$ peut être :

- Attribut *CDATA* : la valeur de l'attribut est une chaîne de caractères prise telle qu'elle est dans le document source.
- Attribut *ENTITY* : la valeur de l'attribut est le nom d'une ou plusieurs entités non XML
- Attribut *NOTATION* : la valeur de l'attribut est le nom d'une notation déclarée précédemment.
- Attribut énuméré : de la forme (par$_1$, par$_2$, ...), la valeur de l'attribut est parmi les valeurs 'par i'.
- Attribut *ID* : Il représente un identifiant pour l'élément, qui doit être unique dans le document.
- Attribut *IDREF* : Il représente une référence à un *ID* se trouvant dans le même document.
- *NMTOKEN* : Il signifie que l'attribut prend comme valeur un nom symbolique quelconque formé de caractères alphanumériques.

Signalons en outre qu'un attribut peut également être une liste de *ID, IDREF, NMTOKEN* ou *ENTITY* séparés par des espaces, grâce aux types *IDS, IDREFS, NMTOKENS* et *ENTITIES* respectivement.

Occurrence$_i$ peut prendre une de ces valeurs:
- ' valeur' ou ' *# DEFAULT* valeur' : la valeur par défaut de l'attribut.
- '*#REQUIRED* ' : la valeur de l'attribut doit impérativement apparaître.
- '*#IMPLIED*' : la présence de l'attribut est facultative dans le document.
- '*#FIXED* valeur': l'attribut doit prendre la valeur 'valeur' dans le document.

3.3. Ecriture XML d'un descripteur LWDOS : le descripteur XLWDOS

3.3.1. Description des partitions

Une partition est représentée par l'élément <P>. Cette balise a un attribut : 'num' de valeur numérique qui représente le numéro de la partie. Sachant que les parties sont numérotées à partir du haut vers le bas et de gauche à droite [21].
Ceci nous donne le format suivant : <P num='x'> description de la partie x </P>.

La description d'une partie comporte deux éléments : <L>, qui nous donne la description de la frontière gauche, et <R> la frontière droite. L'ordre est imposé entre les éléments <L> et <R>. Ceci nous donne le format suivant : <P num='x'>

<L> description frontière gauche </L>
<R> description frontière droite </R>
</P>

La description des frontières d'une partition comporte la description des différents segments qui la constituent. Ces segments peuvent êtres :

- une ligne droite, représentée par l'élément <LN>. Cette balise a deux attributs : 'inclin' qui représente l'inclinaison de la droite par rapport à l'horizon et 'length' qui représente sa longueur.
Ceci nous donne le format suivant : <LN inclin='xx' length='yy' />.

- une courbe concave représentée par l'élément <CC>, ou bien convexe représentée par <CV>. Ces balises ont trois attributs : 'inclin', 'length' et 'degre' pour représenter respectivement : l'inclinaison par rapport aux lignes, la longueur et le degré de convexité ou de concavité.
Ceci nous donne le format suivant : <CC inclin='xx' length='yy' degre='zz' />
<CV inclin='xx' length='yy' degre='zz' />.

Ces trois éléments peuvent apparaître à l'intérieur de <L> ou <R> dans n'importe quel ordre et n'importe quelle occurrence selon le cas.

3.3.2. Description des lignes de Jonction et de Disjonction

Une ligne de jonction est représentée par l'élément <J> et une ligne de disjonction par <D> de la façon suivante :

<J> description de la jonction </J>
<D> description de la disjonction </D>.

Sachant qu'une ligne de Jonction ou de Disjonction est composée de trois types de segments : j, w, h (dénotant respectivement les attributs jonction, FreeLow et FreeHigh). Ils sont représentés par les éléments :

- <JN>, pour jonction. Comporte trois attributs : 'numpart1' qui est le numéro de la 1ere partition à laquelle le segment appartient, 'numpart2' le numéro de la seconde partition et 'length' la longueur du segment.
Ceci nous donne : <JN numpart1='x' numpart2='y' length='z' />.

- <W>, pour FreeLow. Comporte deux attributs : 'numpart' qui est le numéro de la partition à laquelle le segment appartient et 'length' sa longueur.
Ceci nous donne : <W numpart='x' length='z' />.

- <H>, pour FreeHigh. Comporte deux attributs : 'numpart' qui est le numéro de la partition à laquelle le segment appartient et 'length' sa longueur.
Ceci nous donne : <H numpart='x' length='z' />.

Ces trois éléments peuvent apparaître à l'intérieur de <J> ou <D> dans n'importe quel ordre et n'importe quelle occurrence selon le cas.

3.3.3. Description des parties composées

Une partie composée est représentée par l'élément <CP> comme suit :
<CP> description de la partie composée </CP>.
Une partie composée contient des partitions liées via des lignes de Jonction ou de Disjonction.

De manière récursive, chaque Partie Composée peut contenir une autre si le document en contient plusieurs :

```
<CP>
    …
    <CP>
        …
        <CP>
            …
        </CP>
        …
    </CP>
    …
</CP>
```

3.3.4. Description de l'ensemble de l'image

La description de l'ensemble des partitions qui constituent l'image est située à l'intérieur de l'élément <XLWDOS>, qui est l'élément racine. C'est à dire que la balise <XLWDOS> s'ouvre une seule fois au début du document et se ferme à la fin. Elle a un attribut 'Name' facultatif qui nous indique le nom de l'image.
Ceci nous donne le format suivant :
<XLWDOS Name='nom_image'> description de la silhouette </XLWDOS>.

La description de la silhouette peut contenir :
- un élément <P>, si l'image contient une seule partition.
- Un élément <CP>, si l'image contient au moins une partie composée.

Remarque : des attributs pouvant contenir des informations diverses sur l'image, peuvent êtres ajoutés.

La figure 3.1 illustre un exemple d'une silhouette. Nous donnons dans ce qui suit d'abord son descripteur écrit suivant le langage LWDOS. Nous donnons ensuite son écriture XML suivant la syntaxe de XLWDOS.

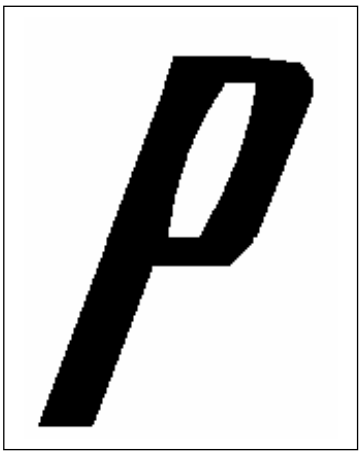

Figure 3.1 Exemple de silhouette

Le descripteur suivant LWDOS :

[[P1 >DJL1 < P2 P3] <JL1 >P4] où :

[[{r 68 12}{cv 0.04 178 3 r 122 9}↓ j 1 2 28 h 1 15 j 1 3 27 ↑ {r 70 75}{cc 0.22 70 75}{cc 0.22 70 75 }{r 90 4 r 69 71}] ↑ j 2 4 28 w 4 16 j 3 4 27 ↓ {r 70 91 r 180 26}{cv 0.13 45 13 r 0 36 r 70 77 }]

Le descripteur suivant XLWDOS :

P.xml

```
<XLWDOS Name = 'logo_p'>
 <CP>
 <CP>
      <P num='1'>
        <L>
          <LN inclin='69'  length='12' />
        </L>
        <R>
          <CV degre='0.04' inclin='176'  length='3'/>
          <LN inclin='122'  length='9' />
        </R>
      </P>
      <D>
        <JN numpart1='1' numpart2='2'  length='28'/>
        <H numpart='1'  length='15'/>
        <JN numpart1='1' numpart2='3'  length='27'/>
      </D>
      <P num='2'>
        <L>
          <LN inclin='70'  length='75'/>
        </L>
        <R>
          <CC degre='0.22'  inclin='70'  length='75'/>
        </R>
      </P>
      <P num='3'>
        <L>
          <CC degre='0.22'  inclin='70'  length='75'/>
        </L>
        <R>
          <LN inclin='90'  length='4' />
          <LN inclin='69'  length='71'/>
        </R>
      </P>
   </CP>
      <J>
        <JN numpart1='2' numpart2='4'  length='28'/>
        <W numpart='4'  length='16'/>
        <JN numpart1='3' numpart2='4'  length='27'/>
      </J>
      <P num='4'>
        <L>
          <LN inclin='70'  length='91' />
          <LN inclin='180'  length='26' />
        </L>
        <R>
          <CV degre='0.13' inclin='45'  length='13'/>
          <LN inclin='0'  length='36' />
          <LN inclin='70'  length='77'/>
        </R>
      </P>
   </CP>
</XLWDOS>
```

3.3.5. Récapitulatif des éléments qui constituent un descripteur XLWDOS

Elément XML	Correspondance LWDOS	Attributs	Eléments fils
XLWDOS	La racine	Nom	CP P
CP	ComposedPart		P CP J D
P	Part	Num	L R
L	LetfBoundary		CV CC LN
R	RightBoundary		CV CC LN
LN	r	length inclin	
CV	cv	length inclin degre	
CC	cc	length inclin degre	
J	Junction		W H JN
D	Disjunction		W H JN
W	h	numpart length	
H	w	numpart length	
JN	j	numpart1 numpart2 length	

3.3.6. DTD du document LWDOS

La DTD (Document Type Definition) suivante définie tous les éléments d'un document XLWDOS, leurs contenus, leurs attributs et les valeurs de leurs attributs comme détaillé plus haut.

Le descripteur XLWDOS doit respecter la DTD (Document Type Definition) suivante :

XLWDOS.dtd

```
<!ELEMENT   XLWDOS (CP|P)>
<!ATTLIST   XLWDOS
            Name  CDATA #IMPLIED >
<!ELEMENT   CP ((((CP ,  P+) | (P+ , CP?) | (P , P+)) , J , P) | ((CP | P) , D , P , P+) )>
<!ELEMENT   P (L,R)>
<!ATTLIST P
            num  CDATA #REQUIRED>
<!ELEMENT   L  (CV|CC|LN)+>
<!ELEMENT   R (CV|CC|LN)+>
<!ELEMENT   LN EMPTY>
<!ATTLIST   LN
            length CDATA #REQUIRED
            inclin CDATA #REQUIRED
<!ELEMENT   CV EMPTY>
<!ATTLIST   CV
            length CDATA #REQUIRED
            inclin CDATA #REQUIRED
            degre CDATA #REQUIRED>
<!ELEMENT   CC EMPTY>
<!ATTLIST   CC
            length CDATA #REQUIRED
            inclin CDATA #REQUIRED
            degre CDATA #REQUIRED>
<!ELEMENT   J (W|H|JN)+>
<!ELEMENT   D (W|H|JN)+>
<!ELEMENT   W EMPTY>
<!ATTLIST   W
            numpart CDATA #REQUIRED
            length CDATA #REQUIRED>
<!ELEMENT   H EMPTY>
<!ATTLIST   H
            numpart CDATA #REQUIRED
            length CDATA #REQUIRED>
<!ELEMENT   JN EMPTY>
<!ATTLIST   JN
            numpart1 CDATA #REQUIRED
            numpart2 CDATA #REQUIRED
            length CDATA #REQUIRED
```

3.4. Conclusion

XLWDOS est une écriture formatée du descripteur LWDOS. Son format est du XML, il est défini par la DTD *XLWDOS.dtd*.
Un descripteur complet, qui contient toutes les informations sur la silhouette, doit être validé par rapport à la DTD où sont décrits les types des attributs, l'obligation de leur présence ou pas, les éléments constitutifs d'une silhouette ainsi que leur ordre d'apparition.
Contrairement à du texte brut non formaté, nous pouvons assuré la possibilité du traitement d'un document XLWDOS **valide**.
De plus, l'écriture XML de XLWDOS permet une extraction de données facile, car il existe plusieurs outils qui permettent un traitement rapide de documents XML.

Chapitre 4 :

Reconstruction de l’image à partir de son descripteur XLWDOS

4.1. Introduction

XLWDOS est un descripteur textuel d'images. Il permet de décrire une silhouette avec du texte formaté tout en vérifiant un ensemble de propriétés : gain en espace mémoire, description de toute l'information, facilité de calcul et d'indexation, … . C'est une méthode de compression de silhouettes.

Dans ce chapitre, nous allons montrer que le processus de description textuelle de l'image peut être inversé. C'est-à-dire, qu'une image peut être reconstruite à partir des données contenues dans un document XLWDOS. Nous montrerons que l'image reconstruite est alors unique et conforme à l'originale. Nous montrerons aussi comment quantifier la qualité du résultat du décodage.

4.2. Unicité de la reconstruction d'une silhouette à partir de son descripteur XLWDOS

Montrons dans cette section que le descripteur XLWDOS contient toute l'information décrite par la silhouette. Nous montrons dans ce qui suit l'unicité de la silhouette reconstruite à partir de son descripteur.

Soient $(a_i, b_i \ldots x_i, y_i, z_i)$ (resp. $(a'_i, b'_i \ldots x'_i, y'_i, z'_i)$) l'ensemble des points de courbure de la frontière gauche (resp. droite) de la partition P_i où a_i , a'_i sont les points de début et z_i, z'_i sont les points de terminaison (voir figures 4.1 et 4.2).

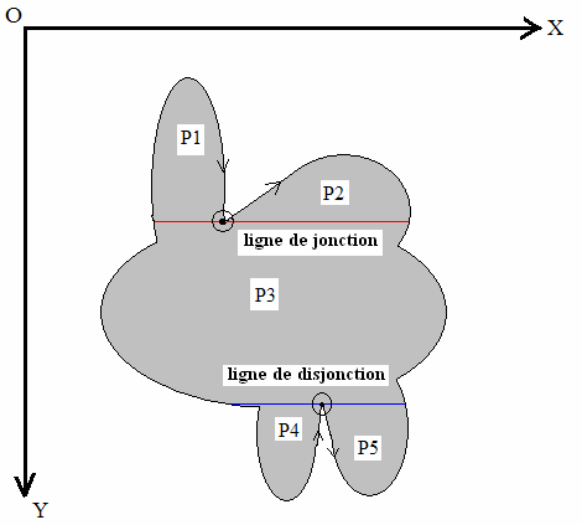

Figure 4.1. Exemple de silhouette

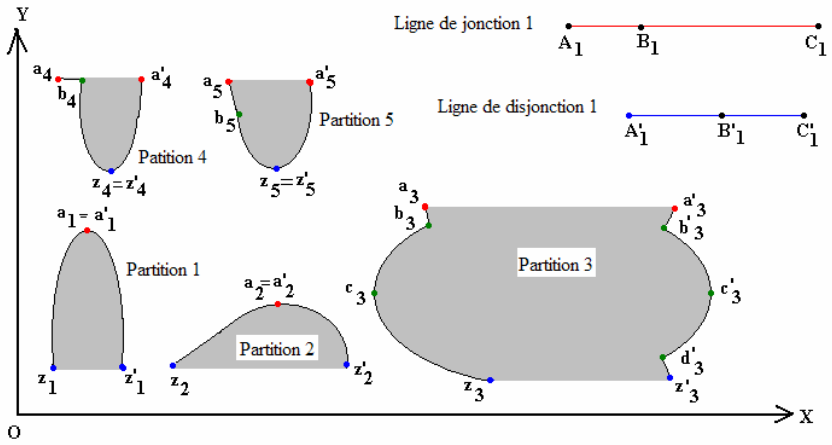

Figure 4.2. Résultat de décomposition d'une silhouette en partitions est lignes de partitionnement

La partition P_i peut être reconstruite en fixant d'abord le point de terminaison z_i et le dessin du contour ($z_i y_i$) connaissant sa longueur, son angle d'inclinaison et son degré de concavité ou de convexité dans le cas d'un contour courbe.

Chacun des autres contours de la frontière gauche : $y_i x_i$… $c_i b_i$ et $b_i a_i$ seront positionnés en utilisant leur point terminaison calculé et leur description. La frontière droite est reconstruite en suivant le même procédé.

Pour joindre les deux frontières, trois cas sont possibles (voir Figure 4.2):

Cas 1: $a_i = a'_i$ et $z_i = z'_i$ (dans le cas où la silhouette est composée uniquement d'une partition)
Cas 2: $a_i = a'_i$ et $z_i \neq z'_i$ (dans le cas où cette partition est suivie par une ligne de partitionnement)
Cas 3: $a_i \neq a'_i$ et $z_i = z'_i$ (dans le cas où cette partition suit une ligne de partitionnement)
Cas 4: $a_i \neq a'_i$ et $z_i \neq z'_i$ (dans le cas où cette partition est délimitée par deux lignes de partitionnement).

Les frontières gauche et droite ont dans les cas 1, 2, et 3 un point commun et donc peuvent être reconstruites et le résultat correspondra à la partition décrite (voir Figure 4.2). Cependant, dans le cas 4, les deux frontières peuvent être reconstruites mais la distance séparant leurs points début et terminaison ne sont pas connus.
Chacune de ces distances seront calculées comme étant la somme des longueurs des segments qui apparaissent dans la description de la ligne de partitionnement correspondante. La figure 4.3 illustre le résultat du processus de reconstruction des partitions ainsi que des lignes de partitionnement de la silhouette de la figure 4.1. Deux partitions du cas 2, deux partitions du cas 3 et une du cas 4.

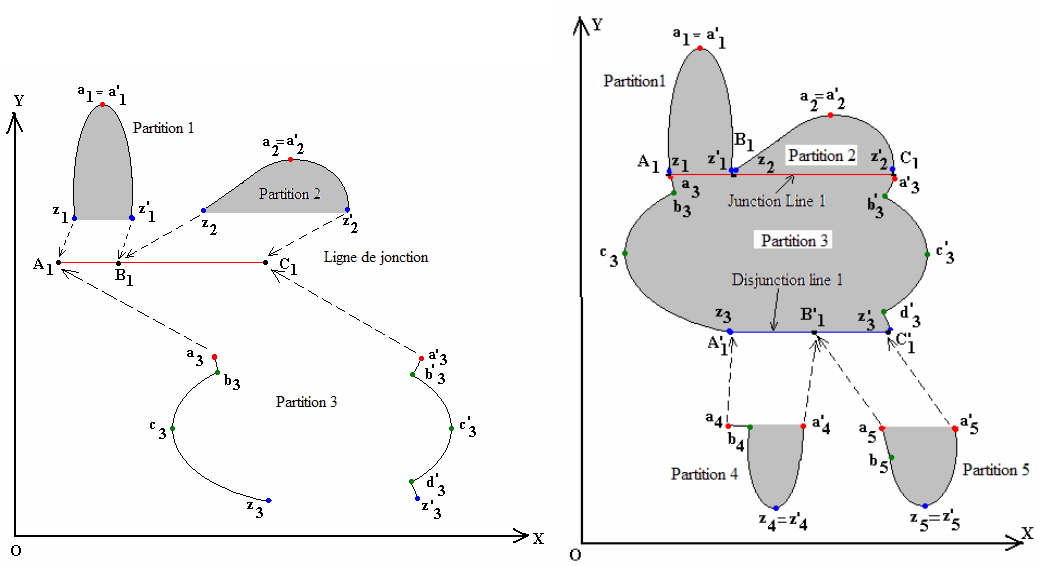

Figure 4.3. Etapes du processus de reconstruction

Les partitions reconstruites en utilisant les attributs des segments qui les constituent (longueur, inclinaison et degré de concavité / convexité) seront positionnées exactement sur les lignes de partitionnement en utilisant leurs descriptions (longueur de chaque segment de la ligne de partitionnement).

4.3. Méthode de reconstruction d'image

En parcourant le descripteur XLWDOS, les opérations suivantes sont effectuées:

- Extraction et sauvegarde des informations relatives à chaque contour élémentaire de la silhouette.
- Extraction et sauvegarde des informations relatives aux lignes de partitionnement (jonction, disjonction) et les différentes partitions qui leur sont liées.
- Reconstruction de l'image.

Le Schéma général de la méthode de reconstruction est décrit par la figure 4.4:

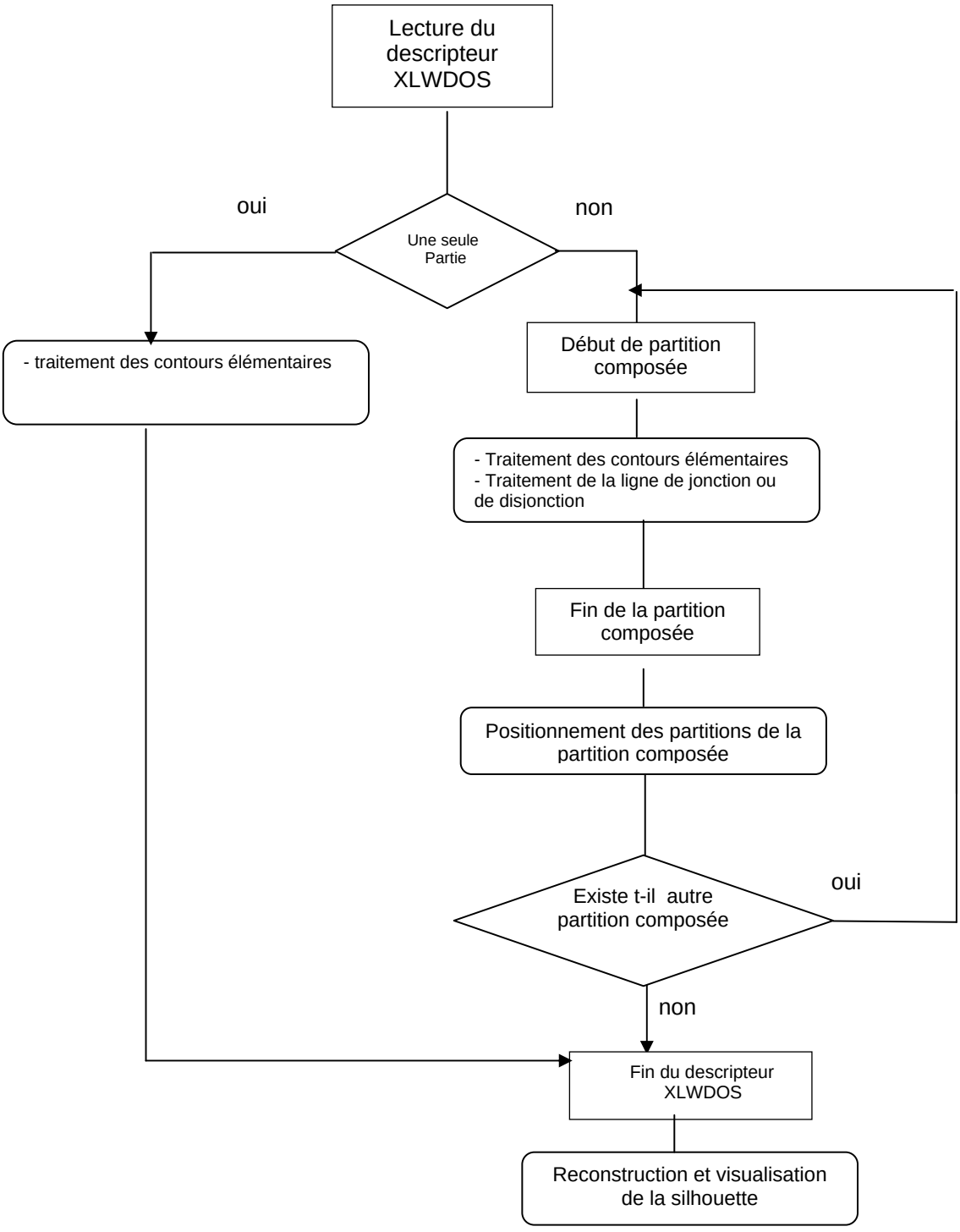

Figure 4.4 : Schéma général de la méthode de reconstruction

4.3.1 Traitement des contours élémentaires:

Pour chaque contour élémentaire C_i, les informations suivantes sont extraites et sauvegardées. Ce traitement s'effectue lorsqu'une balise ouvrante 'LN', 'CC' ou 'CV' est rencontrée:

C_i= (Type, Numéro de partition, longueur, Inclinaison, Degré, Côté, CV, (x_a,y_a), (x_b, y_b), (x_c, y_c)) où:

- Type : défini si le segment est une LN, CV ou CC.
- Numéro partition : est le numéro de la partition à laquelle appartient le segment.
- Longueur, inclinaison : sont la longueur et l'inclinaison du segment
- Degré : est le degré de concavité ou de convexité d'un contour courbé
- Côté : indique si le segment appartient à la frontière gauche ou droite
-CV : bouléen positionné à TRUE si le segment est une courbe *convexe*, à FALSE si elle est *concave.* Cette information est utilisée lors de l'affichage des courbes.
- (x_a,y_a), (x_b, y_b) : sont les coordonnées des extrémités A, B du contour, alors que (x_c,y_c) sont celles de l'extrémité C du contour le plus éloigné de la courbe dans le cas d'un contour courbé.

4.3.2. Traitement des lignes de jonction/disjonction

Une ligne de jonction ou de disjonction est décomposée en segments S_i de type « JN », « W », « H ». Les informations concernant chacun de ces segments sont extraits et sauvegardés.
S_i = (Type, Numéro de partition *High*, Numéro de partition *Low*, Longueur, Numéro de la partition composée)
Où:

-Type : contient le type du segment de partitionnement. « JN », « W » ou « H ».
-NumpartH et NumpartL et Longueur sont les valeurs des attributs *numpart* et *length* contenues à l'intérieur des balises JN, W et H.
- Numéro de CP : contient le numéro de la partie composée courante. Ce numéro est incrémenté à chaque nouvel élément <CP> et décrémenté à sa fermeture </CP>.

4.3.3. Traitement des partitions

Une partition Pi est définie par les informations suivantes:
P_i = (Numéro de la partition, Coordonnées des points de référence gauche et droit: (x_L,y_L), (x_R, y_R)).
Ces points de référence seront utilisés pour positionner la partition relativement à la ligne de jonction ou de disjonction et seront calculés au moment de la reconstruction de la silhouette. Ces points ne correspondent pas forcément aux points de début des frontières gauche et droite qui sont considérés dans le descripteur comme étant des points de référence et l'ensemble des contours élémentaires décrits relativement à ces points.
Ce choix se justifie par le fait que les partitions seront positionnées relativement aux lignes de jonctions et de disjonction.

Ces points de référence sont considérés selon les deux cas suivants :

Cas 1 : La silhouette est composée d'une seule partition

Les points de référence gauche et droit sont identiques. Ils coïncident avec le point de début de la partition (voir figure 4.5).

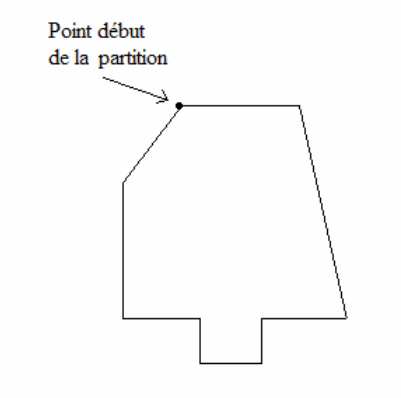

Figure 4.5: Points de début gauche et droit d'une partition

Cas 2 : La silhouette est composée de plusieurs partitions

Dans ce cas, les coordonnées des points de référence partition peuvent être soit des points de début ou de terminaison des deux frontières. Cela dépend de la position de la partition relativement à la ligne de jonction ou disjonction (voir figure 4.6).

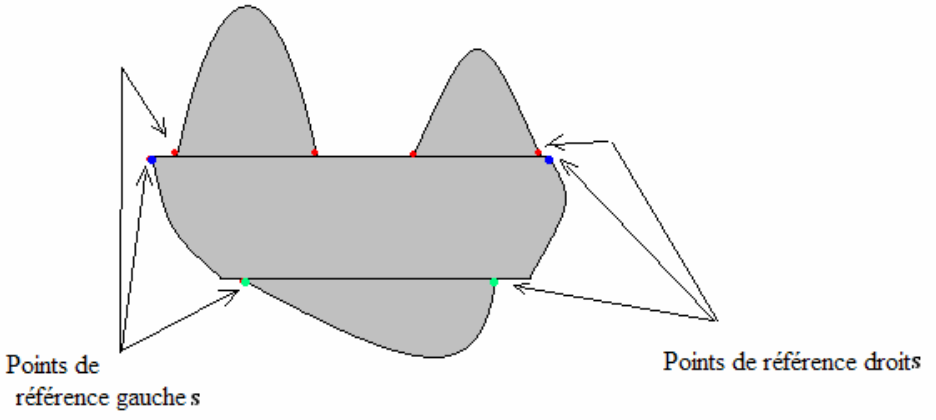

Figure 4.6 Les points de référence d'une partition

Le point de référence gauche (resp. droit) de la partition est le point voisin vertical du point début du segment correspondant dans la ligne de jonction ou disjonction. La figure 4.7 illustre un exemple d'une partition composée. Les points de référence de chaque partition sont colorés en rouge, alors que les points début et fin de chaque segment de la ligne de partitionnement sont colorés respectivement en jaune et bleu (voir figure 4.7).

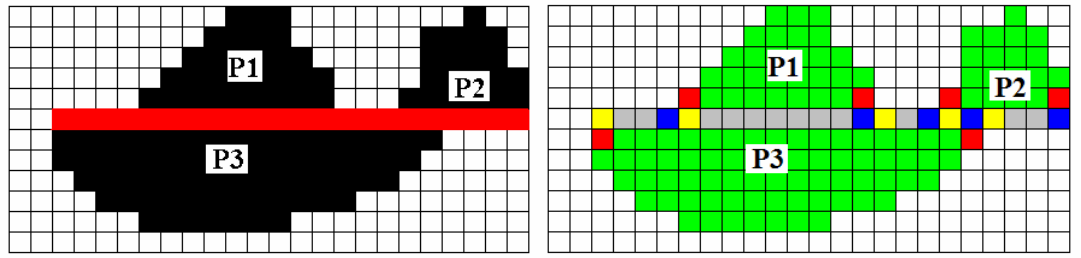

Figure 4.7 A gauche un exemple d'une partition composée, à droite marquage des points de référence des différentes partitions

4.3.4. Traitement des partitions composées

Une partition composée CP_i est définie par une ligne de partitionnement (jonction ou disjonction) et l'ensemble des partitions ou partitions composées en liaison.
$CP_i=(E_{i,1}, E_{i,2}, ..., E_{i,n}, LP_i, E_{i,n+1})$, ou bien $CP_i=(E_{i,1}, LP_i, E_{i,2}, ..., E_{i,n}, E_{i,n+1})$
où
E_i est une partition simple ou partition composée.
LP_i est une ligne de partitionnement.

Suite à la localisation d'une partition composée dans un descripteur XLWDOS, un traitement lui est associé qui consiste à reconstruire ses éléments.

4.3.5 Reconstruction de la silhouette

La reconstruction des partitions composées s'effectuera selon les règles suivantes:

Règle 1:

Pour reconstruire une partition composée d'une silhouette, il est nécessaire et suffisant que l'un de ses éléments soit déjà positionné.

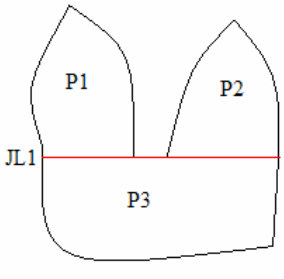

Figure 4.8

Démonstration

Une partition composée est constituée d'un ensemble de partitions et une ligne de partitionnement (voir figure 4.8). Deux cas sont possibles :
- Supposons que la ligne de partitionnement est positionnée dans l'image. Les deux extrémités de chaque segment sont connus et représentent les points de références de chaque partition. Le positionnement des partitions devient donc possible connaissant les positions de ces points de référence.
- Supposons qu'une partition des partitions est positionnée. Ceci implique que ces deux points de référence gauche et droite ainsi que les contours élémentaires sont positionnés. Les deux points de référence de cette partition définissent un des segments de la ligne de partitionnement. Ce segment est donc positionné impliquant le positionnement du reste de segments et des partitions.

Règle 2:

Pour positionner les éléments d'une silhouette, il suffit de positionner un de ses éléments (de préférence une ligne de partitionnement). Le reste de ses éléments (partitions, lignes de partitionnement, partitions composées) seront positionnés par propagation de la règle 1.

Démonstration

Supposons qu'un élément est positionné, à savoir une partition ou une ligne de partitionnement. D'après la règle 1, les éléments de la partition composée définie par la ligne de partitionnement est donc positionné.
Cette partition composée est nécessairement en liaison avec d'autres éléments de la silhouette par le biais d'une de ses partitions déjà positionné. Ceci permet de positionner tous les éléments de la partition composée englobant celle déjà positionnée.
Ce processus de propagation est répété jusqu'au positionnement de tous les éléments de la silhouette.

Règle 3:

La reconstruction d'une silhouette est indépendante du choix du premier élément à positionner.

Démonstration

La démonstration de cette règle a été donnée dans les deux démonstrations précédentes.

4.3.5.1 Amorçage de la reconstruction

Le premier élément à positionner est un élément de la première partition composée rencontrée en parcourant le descripteur de gauche à droite n'ayant pas comme éléments des partitions composées. La ligne de partitionnement est un élément qui permet de faciliter le positionnement des partitions qui sont en liaison.

Pour ce faire, il suffit d'associer l'origine d'un repère (Ouv) de reconstruction au premier pixel de la ligne de partitionnement. Les coordonnées des points de référence de chacune des partitions en liaison avec cette ligne sont obtenues suite à l'exploitation des informations contenues dans la description de ses segments.

Supposons que:

- LP_i est définie par l'ensemble de **n** segment ($S_{i,1}$, $S_{i,2}$, ..., $S_{i,n}$) décrits dans cet ordre
- **m** partitions sont en liaison avec cette ligne LP_i: ($P_{i,1}$, $P_{i,2}$, ..., $P_{i,m}$)
- Chacun des segments $S_{i,k}$ possède les attributs : (type, longueur, Num1,{Num2}), où Num1($S_{i,k}$) et Num2($S_{i,k}$) sont dans l'ensemble ($P_{i,1}$, $P_{i,2}$, ..., $P_{i,m}$)
- Les points de référence d'une partition $P_{i,j}$ sont notés: bL($P_{i,j}$) et bR($P_{i,j}$) où: b , L et R dénotent begin, Left et Right.

Notations :
U_H est une variable pour la sauvegarde de l'abscisse U courante pour les partitions de haut (Heigh)
U_L est une variable pour la sauvegarde de l'abscisse U courante pour les partitions de bas (Low)
NumH est el numéro de la partition *High* liée au segment.
NumL est el numéro de la partition *Low* liée au segment.

Avant de donner l'algorithme, expliquons les différentes actions à effectuer.

- Si le type du segment est « H », alors il peut être au début, au milieu ou à la fin de la partition composée (voir figure 4.9).

L'abscisse U du point de référence gauche de NumH($S_{i,k}$) est égale à U_H, selon les deux cas suivants :

Dans le cas où il commence la ligne de partitionnement (k=1) U_H=0.
Dans les deux autres cas, si NumH($S_{i,k}$)≠NumH($S_{i,k-1}$), l'abscisse de son point de référence gauche est égale à la valeur courante de U_H.

L'abscisse du point de référence droit de NumH($S_{i,k}$) est mise à jour (égale à l'abscisse courante U_H + la longueur du segment courant).

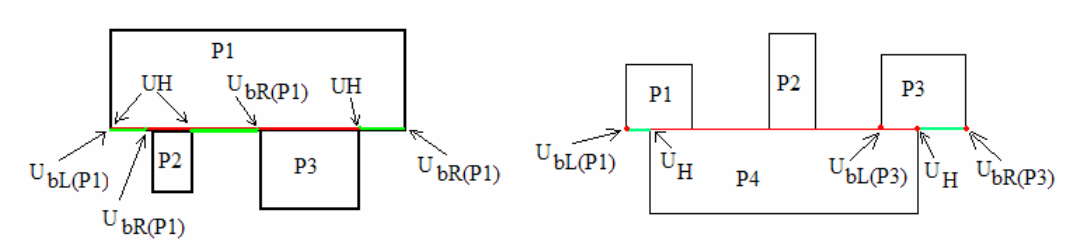

Figure 4.9 Cas de segments ayant l'attribut H

- Si le type du segment est « W », alors il peut être au début, au milieu ou à la fin de la partition composée (voir figure 4.10).

L'abscisse U du point de référence gauche de NumL($S_{i,k}$) est égale à U_L, selon les deux cas suivants :

Dans le cas où il commence la ligne de partitionnement (k=1) U_L=0.
Dans les deux autres cas, si NumL($S_{i,k}$)≠NumL($S_{i,k-1}$), l'abscisse de son point de référence gauche est égale à la valeur courante de UL.

L'abscisse du point de référence droit de NumL($S_{i,k}$) est mise à jour (égale à l'abscisse courante U_L + la longueur du segment courant).

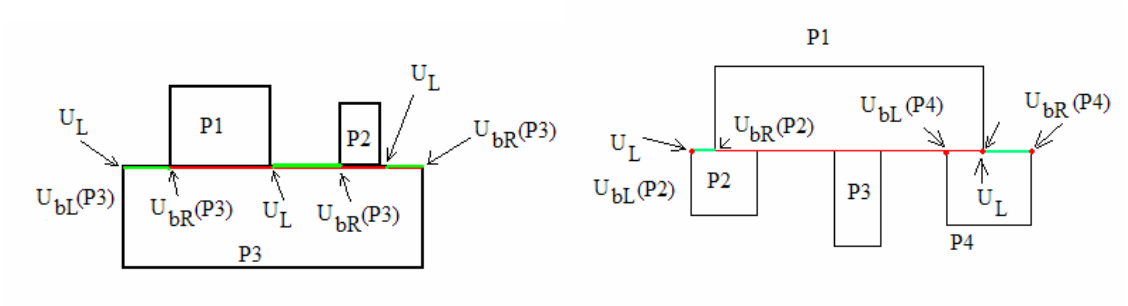

Figure 4.10 Cas de segments ayant l'attribut W

- Si l'attribut du segment est « JN », alors il peut être au début, au milieu ou à la fin de la partition composée (voir figure 4.11).

Les abscisses U des points de référence gauche des deux partitions en liaison sont égales respectivement à U_H et U_L (U_H=U_L=0) dans le cas il commence la ligne de partitionnement (k=1). Dans les deux autres cas :

- Si NumH($S_{i,k}$)≠NumH($S_{i,k-1}$), l'abscisse du point de référence gauche de la partition NumH($S_{i,k}$) est égale à la valeur courante de U_H.
- Si NumL($S_{i,k}$)≠NumL($S_{i,k-1}$), l'abscisse du point de référence gauche NumL($S_{i,k}$)est égale à la valeur courante de U_L.

Les abscisses U des points de référence droits de NumL($S_{i,k}$) et NumH($S_{i,k}$) sont mises à jour (égales à l'abscisse courante U_L + la longueur du segment courant).

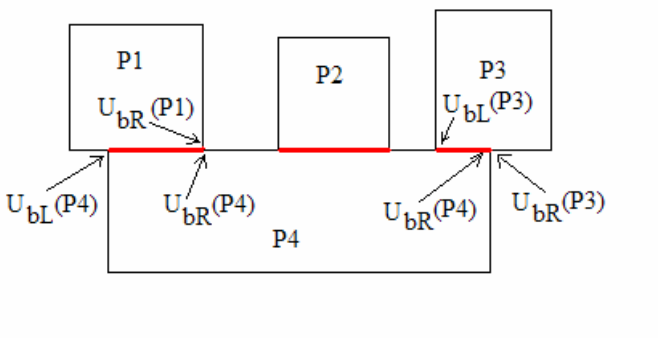

Figure 4.11 Cas ou l'attribut du segment est JN

Algorithme: Amorçage de la reconstruction

Begin

$U_H=0$

$U_L=0$

Select case Type($S_{i,k}$) **of**

Case "H"

- **If** ((k=1) ou ($Num1(S_{i,k}) \neq Num1(S_{i,k-1})$))
  Then $U_{bL}(Num1(S_{i,k})) = U_H$
- $U_H=U_H+Longueur(S_{i,k})$
- $U_{bR}(Num1(_{Si,k}))= U_H - 1$
- $U_L=U_L+Longueur(S_{i,k})$

Case "JN"

- **If** ((k=1) ou ($Num1(S_{i,k}) \neq Num1(S_{i,k-1})$))
  Then $U_{bL}(Num1(S_{i,k})) = U_H$
- $U_H=U_H+Longueur(S_{i,k})$
- $U_{bR}(Num1(S_{i,k}))= U_H - 1$
- **If** ((k=1) ou ($Num2(S_{i,k}) \neq Num2(S_{i,k-1})$))
  Then $U_{bL}(Num1(S_{i,k})) = U_L$
- $U_L=U_L+Longueur(S_{i,k})$
- $U_{bR}(Num2(S_{i,k}))= U_L - 1$

Case "W"

- **If** ((k=1) ou ($Num2(S_{i,k}) \neq Num2(S_{i,k-1})$))
  Then $U_{bL}(Num1(S_{i,k})) = U_L$
- $U_L=U_L+Longueur(S_{i,k})$
- $U_{bR}(Num2(S_{i,k}))= U_L - 1$
- $U_H=U_H+Longueur(S_{i,k})$

EndSelectCase

For each $P_{i,j}$

Do

Reconstruire la partition ($P_{i,j}$, $U_{bL}(P_{i,j})$, $U_{bR}(P_{i,j})$)

EndFor

End

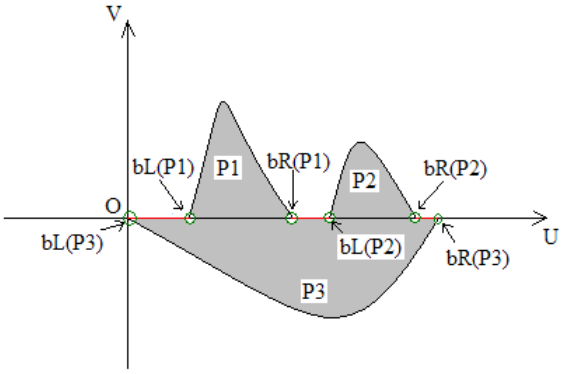

Figure 4.12 Exemple de positionnement de l'élément initial de reconstruction

La figure 4.12 illustre un exemple de positionnement du premier pixel de la ligne de jonction à l'origine du repère de reconstruction (OUV).
La description de la ligne de jonction dans ce cas contient pour chacun des cinq segments, sa longueur et les numéros des partitions en liaison.

4.3.5.2 Propagation de la reconstruction

Ayant une *partie composée* CP_i positionnée, le principe de la propagation consiste à positionner toute autre partition composée CP_{i+1} ayant une partition P_c en commun avec CP_i. Comme illustré par la figure 4.13, la première partition composée positionnée implique le positionnement de la partition P3. La seconde partition composée contient l'élément P3 et elle sera donc positionnée commençant à partir de P3.

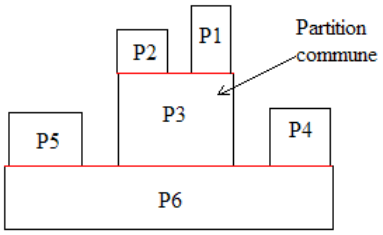

Figure 4.13 : positionnement de la deuxième ligne de jonction

Montrons comment peut-on positionner une partition composée sachant qu'une de ses partitions est déjà positionnée.
Pour ce faire il est nécessaire de connaître les positions des points terminaison des frontières gauche et droite de la partition P_c.

Ces points de terminaison sont calculés suite au positionnement des frontières gauche et droite de la partition P_c. Ce sont les points de la partition déjà positionnée voisins verticaux de la ligne de partitionnement de la partition composée CP_{i+1}.

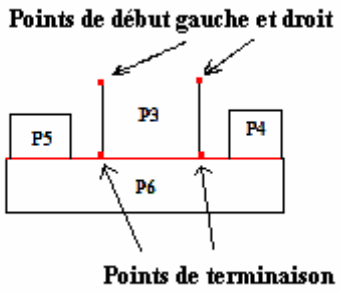

Figure 4.14 : Calcul des points de référence début gauches
et droits des partitions

Ces points permettent de déduire les positions des extrémités du segment correspondant dans la ligne de partitionnement de la partition composée CP_{i+1}, et donc l'extrémité gauche de cette ligne de partitionnement. Donc, U_H et U_L seront initialisés à l'extrémité gauche de la ligne de partitionnement.

En appliquant l'algorithme d'amorçage de reconstruction en il est possible de retrouver les points de référence de chaque partition de la partition composée CP_{i+1}, et donc de les positionner.

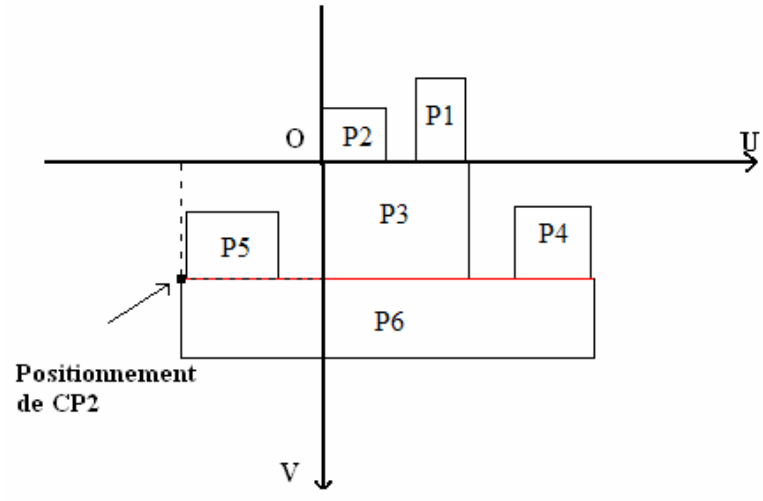

Figure 4.15 : Exemple de positionnement d'une partie
composée par rapport à une autre

4.3.5.3 Calcul des coordonnées des contours élémentaires

A ce stade, nous avons les points de début des frontières de chaque partition et de chaque partie composées de la silhouette
Pour dessiner une frontière, il faut calculer les coordonnées des segments qui la constituent et les coordonnées de l'extrémité du contour le plus éloigné de la courbe dans le cas d'un contour courbé (voir figure 4.18).

- Calcul des coordonnées des segments :

En considérant que chaque segment commence à l'extrémité du segment précédent. On calcule ses coordonnées de fin (x_b, y_b) en utilisant les attributs du segment, c'est à dire, sa longueur et son inclinaison.

Le calcul des coordonnées des segments se fait selon le procédé suivant :
En commençant par le point de début gauche (respectivement droit) on calcule les coordonnées des segments de la frontière gauche (respectivement droite) en balayant la frontière à partir de la ligne de jonction/disjonction vers l'extérieur. C'est à dire, pour les partitions se trouvant dans la partie 'high' de la ligne de jonction/disjonction, le sens est de bas en haut (voir figure 4.16).

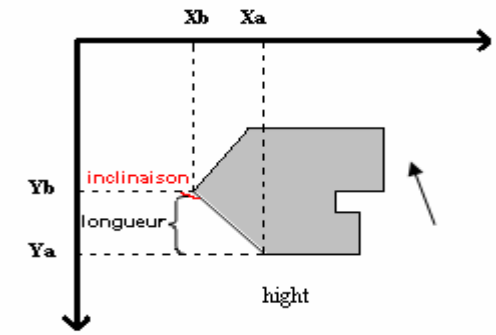

Figure 4.16 : calcul des coordonnées de début et de fin des segments pour une partie *high*

Les coordonnées (X_b,Y_b) du segment S_i sont calculées selon la formule suivante :

$$X_b(S_i) = X_a(S_i) + \frac{longueur(S_i)}{\tan(inclinaison(S_i))}$$

$$Y_b(S_i) = Y_a(S_i) - (longueur(S_i))$$

Pour les partitions se trouvant dans la partie 'low' de la ligne de jonction/disjonction, ainsi que les parties qui ne sont pas liées à des lignes de partitionnement, le sens du balayage est de haut en bas (voir figure 4.17).

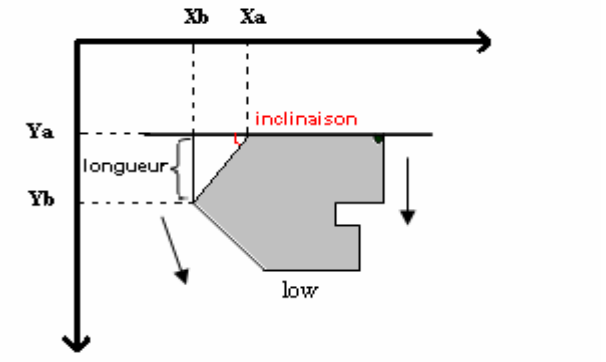

Figure 4.17 : calcul des coordonnées de début et de fin des segments pour une partie *low*

Les coordonnées (X_b,Y_b) du segment S_i sont calculées selon la formule suivante :

$$X_b(S_i) = X_a(S_i) - \frac{longueur(S_i)}{\tan(inclinaison(S_i))}$$

$$Y_b(S_i) = Y_a(S_i) + (longueur(S_i))$$

- Calcul des coordonnées de l'extrémité de la courbure

Les coordonnées du point le plus loin du contour courbé sont nécessaires pour pouvoir le dessiner. Le calcul se fait selon le procédé suivant :

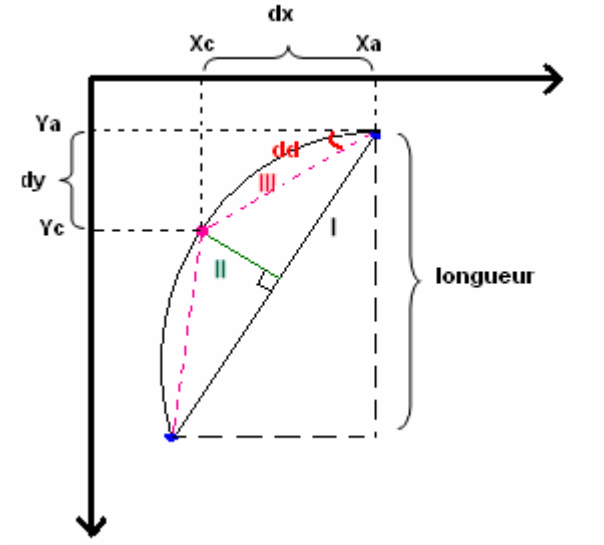

Figure 4.18 : Calcul des coordonnées (Xc,Yc)

Les extrémités des segments CV et CC étant calculées comme pour les segments. Il suffit de calculer l'extrémité de la courbure en utilisant le degré de convexité/concavité, selon les formules suivantes (voir figure 4.18) :

$$l = \frac{longueur(S_i)}{2 * \sin(inclinaison(S_i))}$$

$$ll = degré(S_i) * (2 * l) \qquad // \ car\ le\ degré\ de\ convexité\ ou\ de\ concavité = \frac{ll}{2*l}$$

$$d = \arctan(\frac{ll}{l})$$

$$dd = inclinaison(S_i) - d$$

$$lll = \frac{l}{\cos(d)}$$

$$dy = lll * \sin(dd)$$

$$dx = lll * \cos(dd)$$

Donc:

$$\boldsymbol{X_c(S_i) = X_a(S_i) - dx} \qquad \text{et} \qquad \boldsymbol{Y_c(S_i) = Y_a(S_i) + dy}$$

4.3.6. Dessiner la silhouette

En premier lieu, tous les segments qui constituent la silhouette sont affichés sans les courbures.

Ensuite on affiche les courbures des segments S_i de la manière suivante :

Pour chaque segment courbé S_i
Begin
If (S_i est convexe)
Then
    If (coté(S_i) = 'left') color = 'black'
    Else color = 'white'

Else
    If (coté(S_i) = 'left') color = 'white'
    Else color = 'black'
End

Les deux figures 4.19 et 4.20, montrent les deux étapes de l'affichage d'une silhouette.

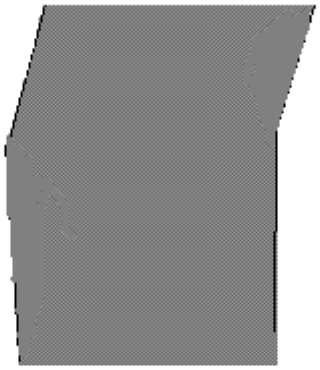

Figure 4.19 1ere étape : joindre les segments

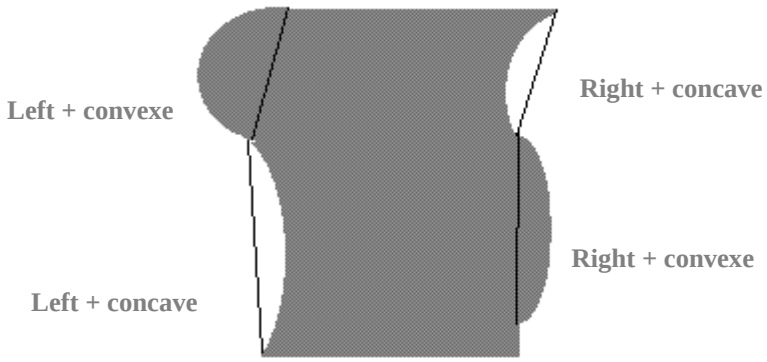

Figure 4.20 2eme étape : dessiner les courbures

4.4. Etude de la qualité de l'image reconstruite

La précision dans la reconstitution des images à partir de leurs descripteurs XLWDOS est liée à la précision avec laquelle ce descripteur a été calculé.

4.4.1. Paramètres du descripteur XLWDOS

Le descripteur XLWDOS utilise trois paramètres pour décrire les contours élémentaires: inclinaison, longueur et degré de concavité/convexité.

- L'inclinaison du segment délimité par les deux points d'extrémité d'une droite ou d'une courbe est calculée par rapport à l'horizontale, dans le sens orthogonal (contraire au déplacement des aiguilles d'une montre) [21].

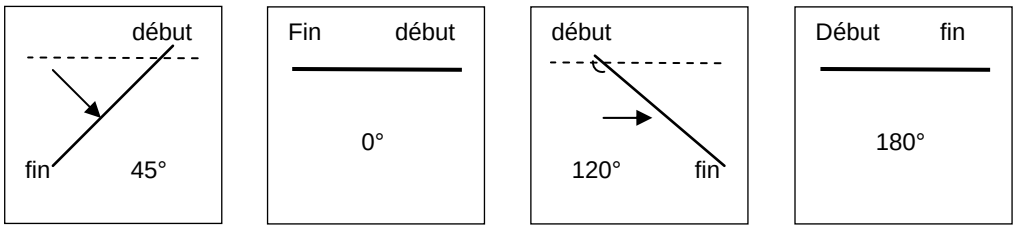

Figure 4.21 : Calcul de l'angle d'inclinaison

- La longueur d'un segment horizontal ou vertical est délimitée par les points de début et de fin de ce segment. Dans le cas où l'inclinaison n'est pas 0° ou 180°, la longueur est donnée par le nombre de lignes qui séparent les deux points d'extrémité [21].

- Le degré de concavité/convexité d'un contour courbé est le rapport de la distance maximale entre les points contours et segment reliant ses extrémités sur la longueur de ce segment

Pour calculer ces paramètres, il faut décomposer chaque frontière en contours élémentaires par l'intermédiaire des « points de courbure ».

L'algorithme de Chetverikov [21] permet de localiser ces points moyennant deux paramètres : l'angle 'α' et la distance 'n'.

Tout au long de la frontière de la silhouette, l'angle 'α' est calculé en prenant 'n' pixels de chaque côté du pixel considéré. Si l'angle est inférieur au paramètre donné, alors le point est alors un point de courbure.

4.4.2. Impact de ces paramètres sur la qualité de reconstruction

- Les paramètres utilisés pour localiser les points de courbure sont de grande importance dans la qualité du descripteur obtenu. En effet plus la distance *n* diminue, et plus l'angle α augmente, les points de courbure deviennent nombreux. Ceci a pour résultat la description avec plus de précision des contours des frontières de chaque partition.

A titre d' exemple, la figure 4.22 illustre une silhouette, le résultat de sa décomposition en contours élémentaires est donné par la figure 4.23 avec différentes valeurs des paramètres *n* et *α*.

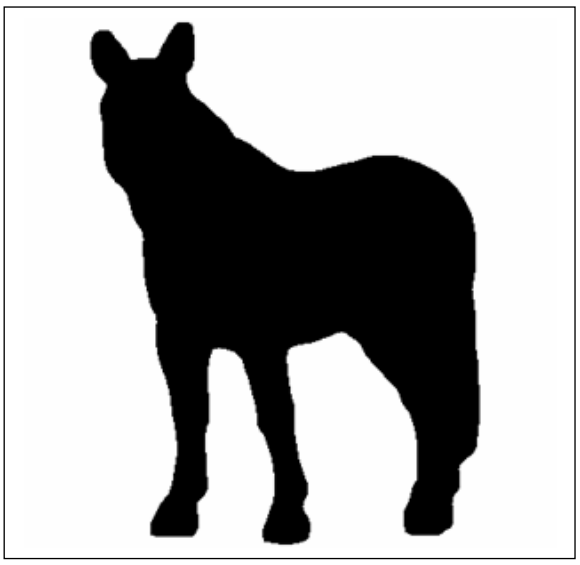

Figure 4.22 Exemple de silhouette

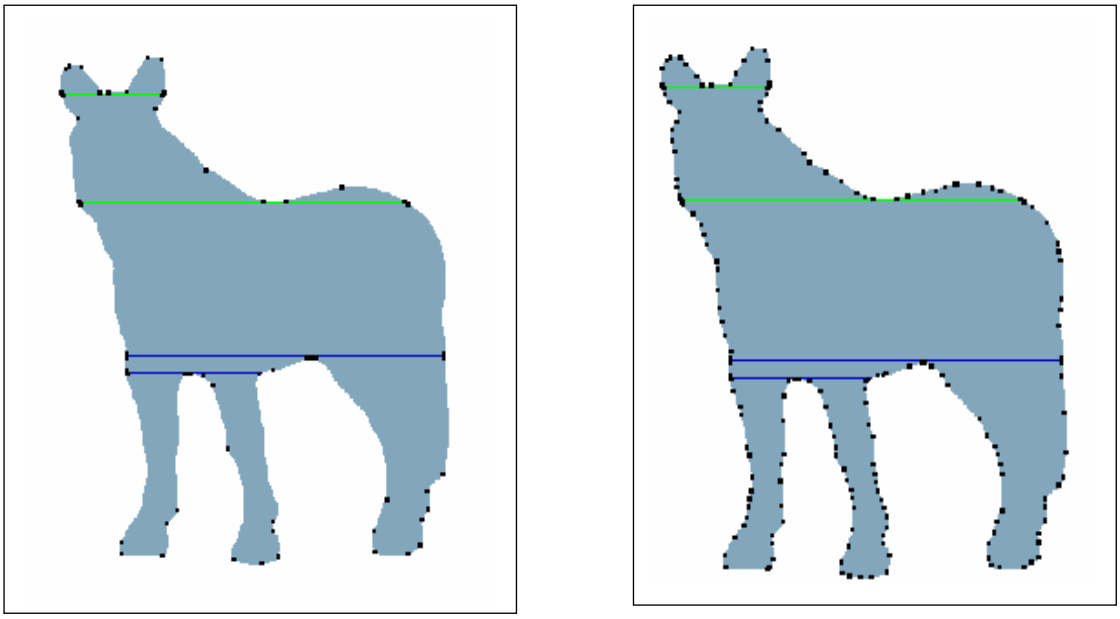

Figure 4.23 Localisation des points de courbure avec (α=150, n=5) pour l'image de gauche et (α=170, n= 4) pour l'image de droite

En conséquence, les deux descripteurs obtenus sont différents en taille et en contenu. Le second contient plus de contours élémentaires et occupe plus d'espace mémoire que le premier descripteur.

4.4.3 Méthode de quantification de la qualité du décodage XLWDOS

Afin de quantifier la qualité des images obtenues, nous superposons l'image originale avec l'image résultat et nous identifions les pixels qui existent sur l'une et pas sur l'autre.
Ceci va nous permettre de calculer le taux de perte : TP = TPP + TPM. Où :

- TPP (Taux de Pixels en Plus) représente le quotient du nombre de pixels en plus dans l'image résultat par le nombre de pixels dans l'image originale.

$$TPP = (\frac{NbPixelPlus}{NbPixel\,\mathrm{Im}ageOriginale}) * 100$$

- TPM (Taux de Pixels en Plus) représente le quotient du nombre de pixels en moins dans l'image résultat par le nombre de pixels dans l'image originale.

$$TPM = (\frac{NbPixelMoins}{NbPixel\,\mathrm{Im}ageOriginale}) * 100$$

Ces taux sont exprimés en pourcentage.

Le TP d'une silhouette sera la somme du TPP et du TPM de cette silhouette.

Afin que deux partitions puissent être comparées, il faut un point de référence pour chacune. Les deux images seront superposées par rapport à ces points de référence.
On fixe P1 et on décale les pixels dans P2 par rapport à la ligne des ordonnées (étape1), ensuite par rapport à la ligne des abscisses (étape 2).

La figure suivante illustre ces deux étapes :

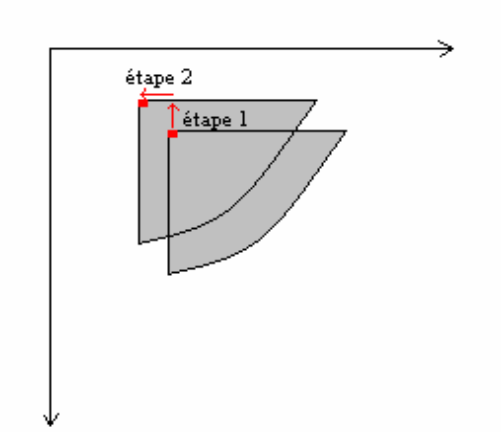

Figure 4.24 Positionnement des deux images à comparer

4.5. Conclusion

Dans ce chapitre nous avons présenté la méthode développée pour la reconstruction de la silhouette à partir de son descripteur XLWDOS.
Dans le chapitre suivant, nous présentons les résultats obtenus sur des images de synthèse et réelles. La quantification de la qualité de l'image reconstruite nous permettra de proposer des perspectives à ce travail et les possibilités d'application de ce format d'images.

Chapitre 5 :

Validation expérimentale

5.1 Introduction

Nous avons conçu pour la reconstruction d'images un outil qui décode les descripteurs XLWDOS et visualise les images correspondantes. Cette application est basée sur la méthode de reconstruction d'images détaillée dans le chapitre 4.

L'outil développé est une classe Java nommée : **XLWDOSImg**. La classe **XLWDOSImg** utilise l'API SAX pour traiter le document XML, puis affiche l'image dans une fenêtre avec un titre de fenêtre qui correspond au nom de l'image. Cette information est tirée du descripteur XML.

Les fonctions de **XLWDOSImg** sont :

Méthodes	Description
String **getName()**	retourne le nom de l'image à traiter
Int **getLength()**	retourne la longueur de la silhouette
Int **getNumCP()**	retourne le nombre de *partitions composées*
Int **getNumP()**	retourne le nombre de *partitions*
Int **getNumJ()**	retourne le nombre de *lignes de jonction*
Int **getNumD()**	retourne le nombre de *lignes de disjonction*
XLWDOSImg (String nom_fichier_XML)	C'est le constructeur de la classe XLWDOSImg. Il crée un parseur SAX et lui donne en entrée le fichier XLWDOS pour le traiter et enregistrer les informations nécessaires à la reconstruction de l'image dans les classes : *segment*, *partie* et *junction*.
DrawXLWDOS(segment S, partie P, junction J)	Cette fonction utilise les fonctions précédentes pour dessiner la silhouette à partir des segments qui la constituent.

Les attributs tels que le nom de l'image et la longueur du descripteur sont extraits du descripteur XLWDOS. Par contre les informations telles que le nombre de parties ou de lignes de jonction sont calculées.

5.2. L'environnement de développement

5.2.1 L'API SAX [30]

Simple API for XML est une API basée sur un modèle événementiel, cela signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML.
Une application utilisant SAX crée un parseur qui contient plusieurs gestionnaires (handlers). Les quatre principaux types de *handlers* sont [28]:

a) Gestionnaire de Contenu : Le *ContentHandler* est chargé des événements lui permettant d'effectuer des opérations selon le type d'élément rencontré.
Voici les 5 évènements détectés par SAX ainsi que les méthodes qui sont appelées :
- Détection d'une balise de début : startElement()
- Détection d'une balise de fin : endElement()
- Détection de données entre deux balises : characters()
- Début du traitement du document : startDocument()
- Fin du traitement du document : endDocument()

b) Gestionnaire d'Erreurs : Le *ErrorHandler* va traiter les trois types d'erreurs possibles lors du parsing : les erreurs simples, les erreurs fatales et les warnings :
- Erreur fatale : Cette erreur ne peut être récupérée. C'est le cas par exemple pour un document mal formé. Après ce type d'erreur, le parseur SAX arrête son travail.
- Erreur simple : Cette erreur peut être récupérée, c'est à dire que le parseur SAX peut continuer à traiter le reste du document XML. Ce genre d'erreur peut se produire lors d'une violation d'une contrainte imposée par la DTD ou le schéma.
- Warning : C'est un simple avertissement. Après cela, le parseur SAX continue le parsing du document.

c) Gestionnaire de DTD : Le *DTDHandler* (Document Type Definition) gère les événements relatifs aux DTD.

d) Gestionnaire d'entités externes : L'*EntityResolver* est chargé de gérer les entités externes, en fournissant une InputSource adéquate.

5.2.2 JAVA [15, 2]

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. Il existe 2 types de programmes en Java : les applets et les applications. Une application autonome est une application qui s'exécute sous le contrôle direct du système d'exploitation. Une applet est une application qui est chargée par un navigateur et qui est exécutée sous le contrôle d'un plug in de ce dernier.
Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

Java est interprété :

Le source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java : la Java Virtual Machine (JVM). Ce concept est à la base du slogan de Sun pour Java : WORA (Write Once, Run Anywhere : écrire une fois, exécuter partout). En effet, le byte code, s'il ne contient pas de code spécifique à une plate-forme particulière peut être exécuté et obtenir quasiment les mêmes résultats sur toutes les machines disposant d'une JVM.

Java est indépendant de toute plate-forme :

Il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du byte code.

Java est économe :

Le pseudo code a une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution. La communauté Java est très productive car elle regroupe :
- Sun, le fondateur de Java.
- le JCP (Java Community Process) : c'est le processus de traitement des évolutions de Java dirigé par Sun.Chaque évolution est traitée dans une JSR (Java Specification Request) par un groupe de travail constitué de différents acteurs du monde Java
- des acteurs commerciaux dont tous les plus grands acteurs du monde informatique excepté Microsoft
- la communauté libre qui produit un très grand nombre d'API et d'outils pour Java.

5.3 Reconstruction d'images avec XLWDOSImg

Afin de valider la méthode proposée et de quantifier la qualité de reconstruction, nous avons réalisé un ensemble de tests de reconstruction de diverses images à partir de leurs descripteurs XLWDOS en utilisant l'application XLWDOSImg.

La première tache réalisée par l'application est le contrôle de validité du descripteur XLWDOS. Pour ce faire, nous avons développé un module de gestion d'erreurs.

5.3.1. Gestion des erreurs

En plus du *Contenthandler* (voir § 5.2.1), qui sert à gérer le contenu d'un document XML, XLWDOSImg implémente aussi le *ErrorHandler,*le *DTDHandler* et L'*EntityResolver* qui lui permettent respectivement de charger l'entité externe qui est la DTD, gérer les évènements qui lui sont liés et traiter les erreurs.

Avant de traiter le document XML pour afficher l'image, XLWDOS vérifie d'abord que c'est un document valide, qui respecte la syntaxe XML et qu'il est conforme à la DTD définissant la structure d'un descripteur XLWDOS.
Si le descripteur ne respecte pas la syntaxe XML ou celle de XLWDOS, l'application retourne une erreur en indiquant le numéro de la ligne et de la colonne dans le fichier XML.

Comme exemple d'un descripteur XLWDOS contenant des erreurs soumis à XLWDOSImg, nous obtenons les commentaires suivants :

poly1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DXLWDOS SYSTEM "file:XLWDOS.dtd" >
<DXLWDOS Name = 'sample02'>
<P num='1'>
<L>
<LN inclin='90' Length='18'/>
<LN inclin='180' Length='39'/>
<LN inclin='90'        />
<LN inclin='180' Length='19'/>
</L>
<R>
<LN inclinaison='180'  Length ='97'/>
<LN inclin='90' Length='18'/>
<LN inclin='0' Length='39'/>
<LN inclin='90' Length='94'/>
</P>
</DXLWDOS>
```

Ce descripteur contient plusieurs erreurs :

- Concernant la syntaxe XLWDOS : l'attribut *Length* est absent d'un élément LN et un attribut *inclin* est remplacé par *inclinaison*.

- Concernant la syntaxe XML : une balise de fermeture </R> manque.
Le parseur qu'implémente XLWDOSImg nous renvoie les erreurs de la manière suivante (voir figure 5.1) :

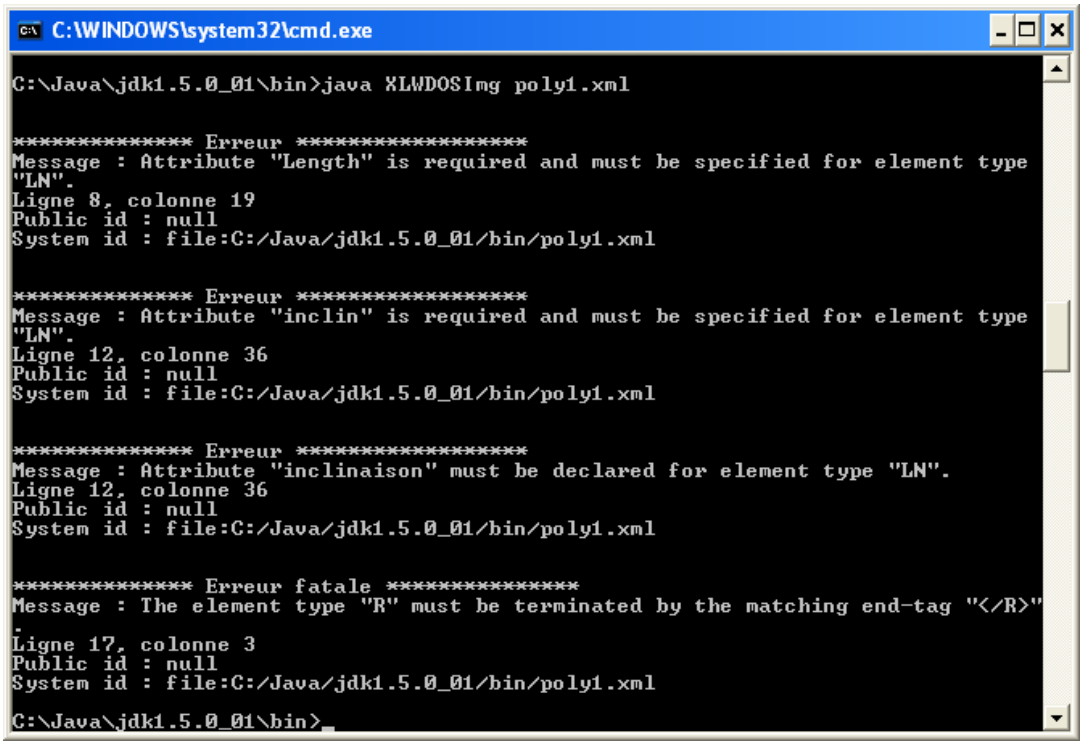

Figure 5.1 : Gestion des erreurs

Il est primordiale que le descripteur soit valide et respecte la syntaxe XLWDOS. Ceci nous assure la disponibilité de toutes les informations nécessaires à la reconstruction des images.

5.3.2. Reconstitution de silhouettes de synthèse

a) Silhouette polygonale à une partition

La figure 5.2 illustre l'image originale à gauche et l'image reconstruite à droite :

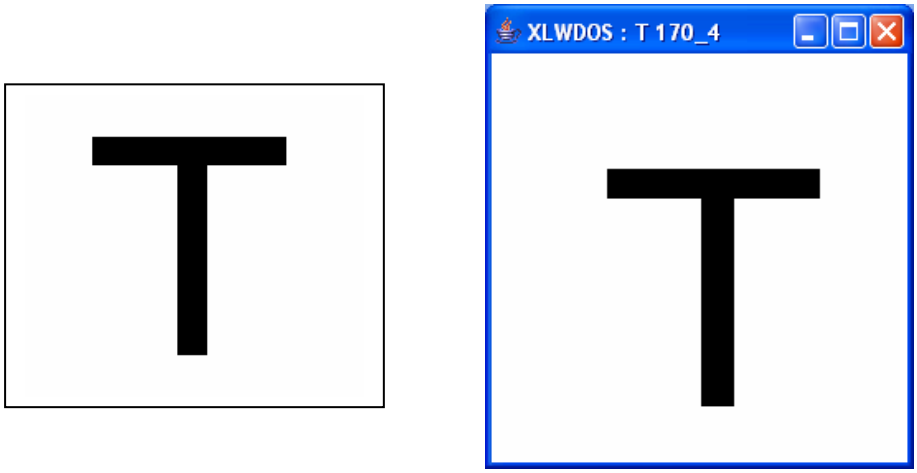

Figure 5.2 : Image avec une seule partition

b) Silhouette polygonale à plusieurs partitions

La figure 5.3 illustre deux images originales à gauche et les images reconstruites à droite :

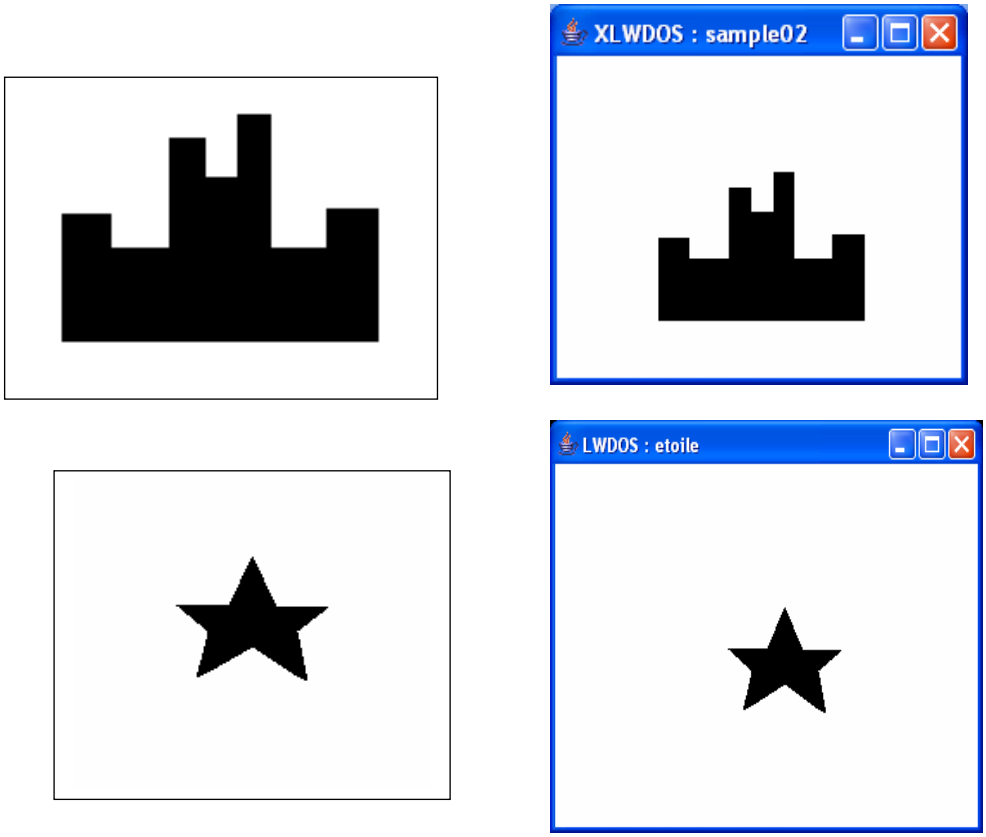

Figure 5.3 : Silhouettes avec plusieurs partitions

c) Silhouette avec courbure à une partition

Dans le cas où le contour externe de la forme a des portions courbées, la description XLWDOS se fait en fixant les paramètres α et n. La reconstruction dépend alors de la précision de la description. Nous donnons ci-après quelques résultats obtenus sur des silhouettes de synthèse.

Exemple1 :

La figure 5.4 illustre un exemple simple d'une silhouette courbée. Les figures 5.5 et 5.6 illustrent les silhouettes reconstruites en utilisant différentes valeurs des paramètres α et n de l'algorithme de Chetverikov.

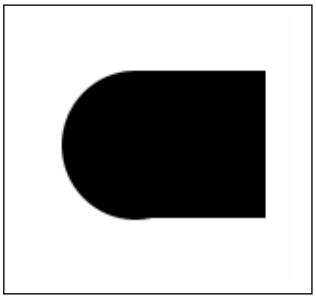

Figure 5.4: la silhouette originale « img01 »

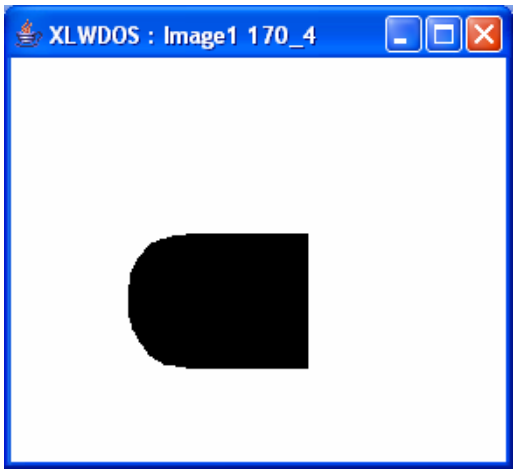

Figure 5.5 : Silhouette reconstruite avec α = 170 et n = 5

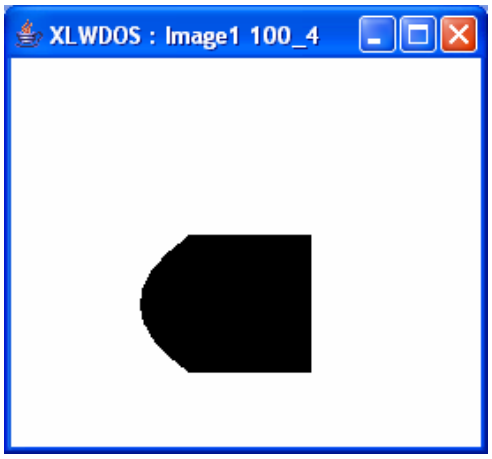

Figure 5.6 : Silhouette reconstruite avec α = 160 et n = 5

Exemple 2 :

La figure 5.7 illustre un autre exemple simple d'une silhouette courbée. Les figures 5.8 et 5.9 illustrent les silhouettes reconstruites en utilisant différentes valeurs des paramètres α et n.

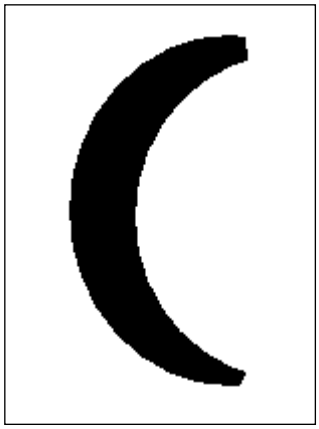

Figure 5.7 : la silhouette originale « croissant »

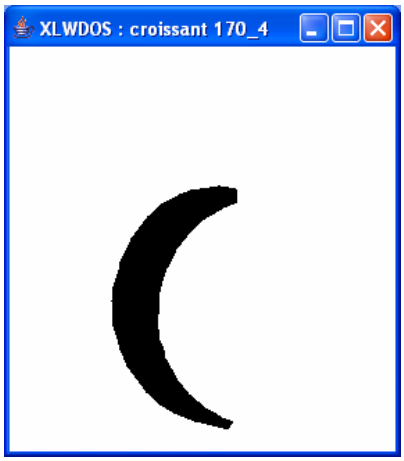

Figure 5.8 : Silhouette reconstruite avec α = 170 et n = 4

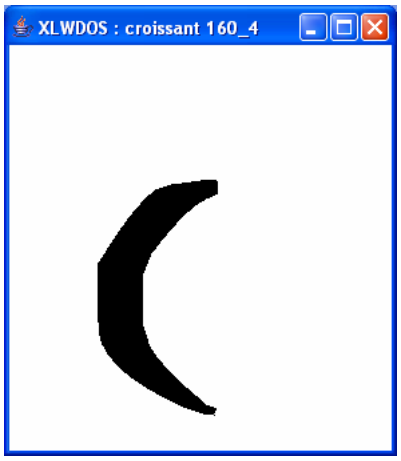

Figure 5.9 : Silhouette reconstruite avec α = 160 et n = 4

d) Silhouette avec courbure à plusieurs partitions composées

La figure 5.10 illustre l'image originale à gauche et l'image reconstruite à droite :

Figure 5.10 : Image avec courbures et plusieurs partitions composées (reconstruction avec α = 160 et n = 4)

Nous avons utilisé pour la validation de l'approche proposée un ensemble d'images de logos industriels et un ensemble d'images réelles prises de la base de données images ETH80 de B. Leibe and B. Schiele [4].

5.3.3. Reconstitution de silhouettes d'images de logos industriels

Exemple1 :

La figure 5.11 illustre un premier exemple de logo industriel. Nous avons extrait les différentes silhouettes contenues dans ce logo et nous avons calculé leurs descripteurs XLWDOS. Les silhouettes reconstruites sont illustrées par la figure 5.12.

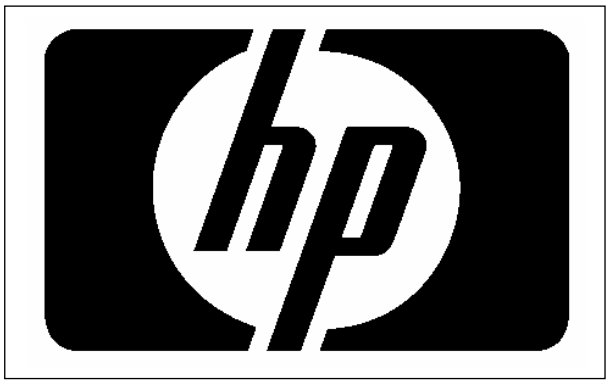

Figure 5.11. Exemple de logo industriel « hp »

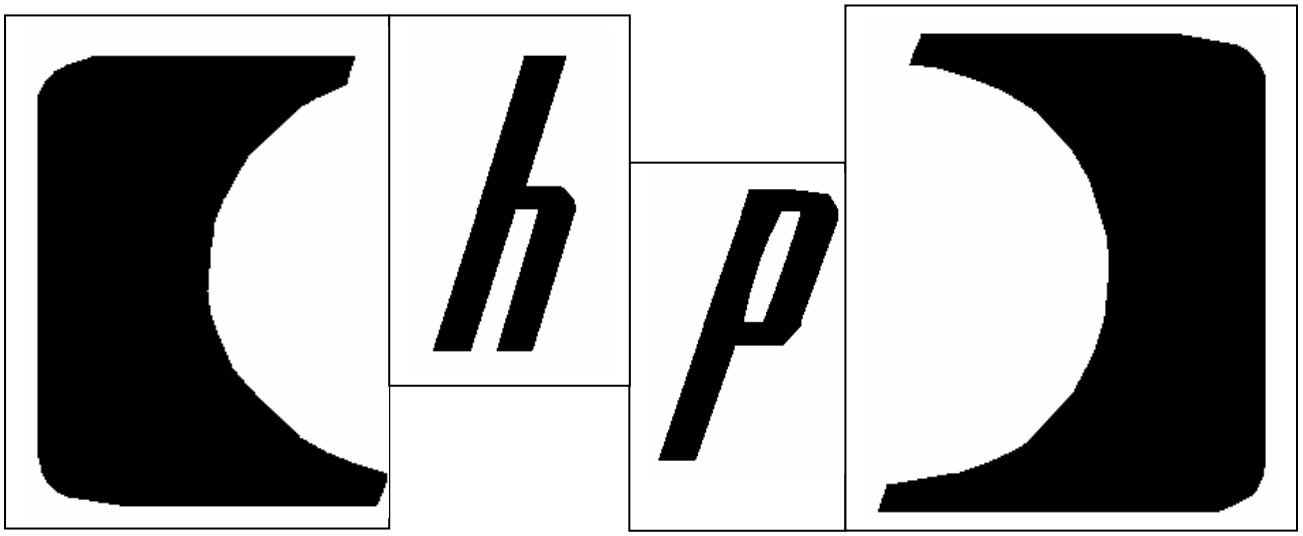

Figure 5.12. Les silhouettes reconstruites avec α = 160 et n = 4

Exemple2

La figure 5.13 illustre un second exemple de logo industriel. Les silhouettes reconstruites sont illustrées par la figure 5.14.

Figure 5.13. Exemple de logo industriel « pepsi »

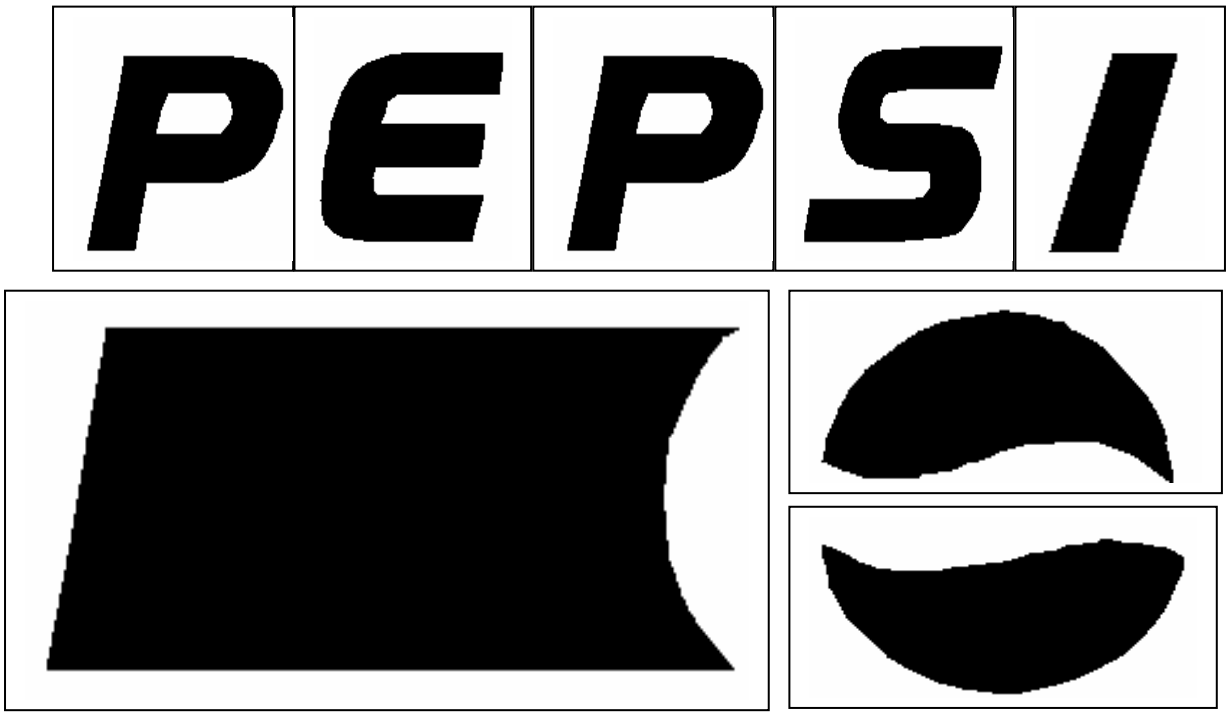

Figure 5.14. Les silhouettes reconstruites avec $\alpha = 160$ et $n = 4$

5.3.4. Reconstitution de silhouettes d'images réelles

Nous présentons le résultat de la reconstruction de quelques silhouettes d'objets réelles.

Exemple1 :

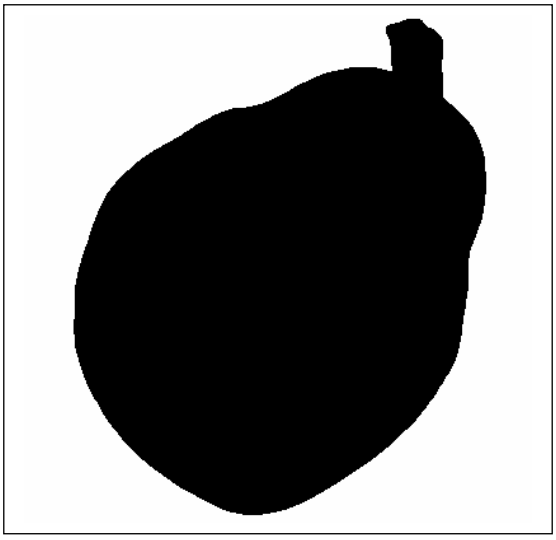

Figure 5.15: la silhouette originale « poire »

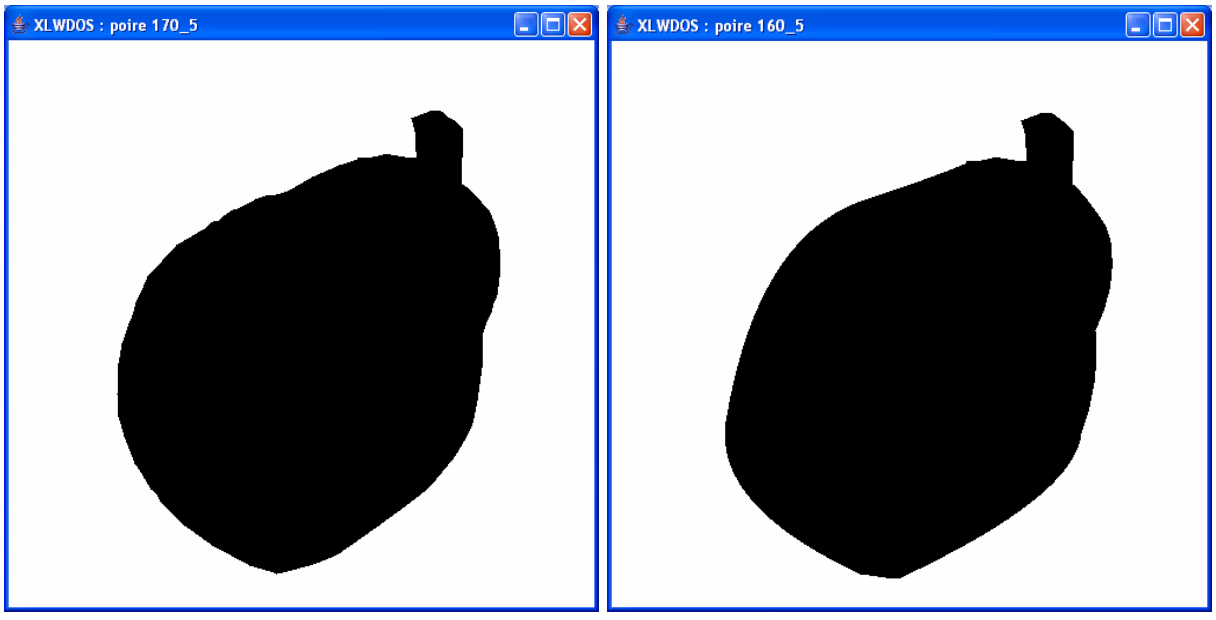

Figure 5.16. Reconstruction de « poire » avec (α = 170, n = 5) et (α = 160, n = 5)

Exemple2 :

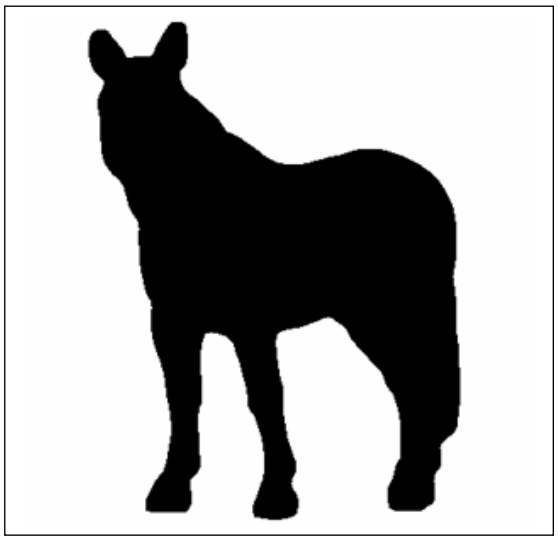

Figure 5.17 : la silhouette originale « cheval »

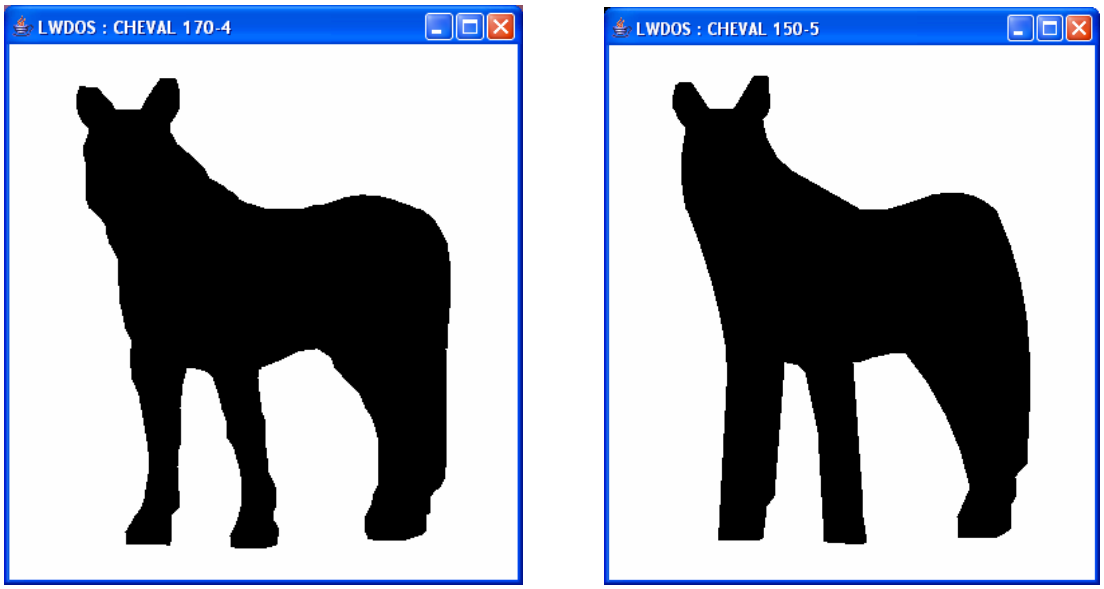

Figure 5.18 : Reconstruction de « cheval » avec (α = 170, n = 4) et (α = 150, n = 5)

Exemple3 :

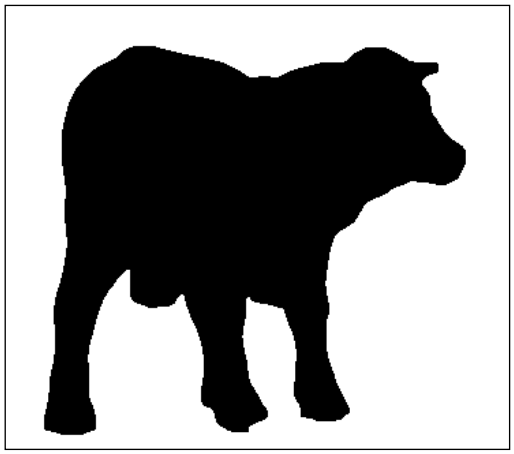

Figure 5.19: la silhouette originale « vache »

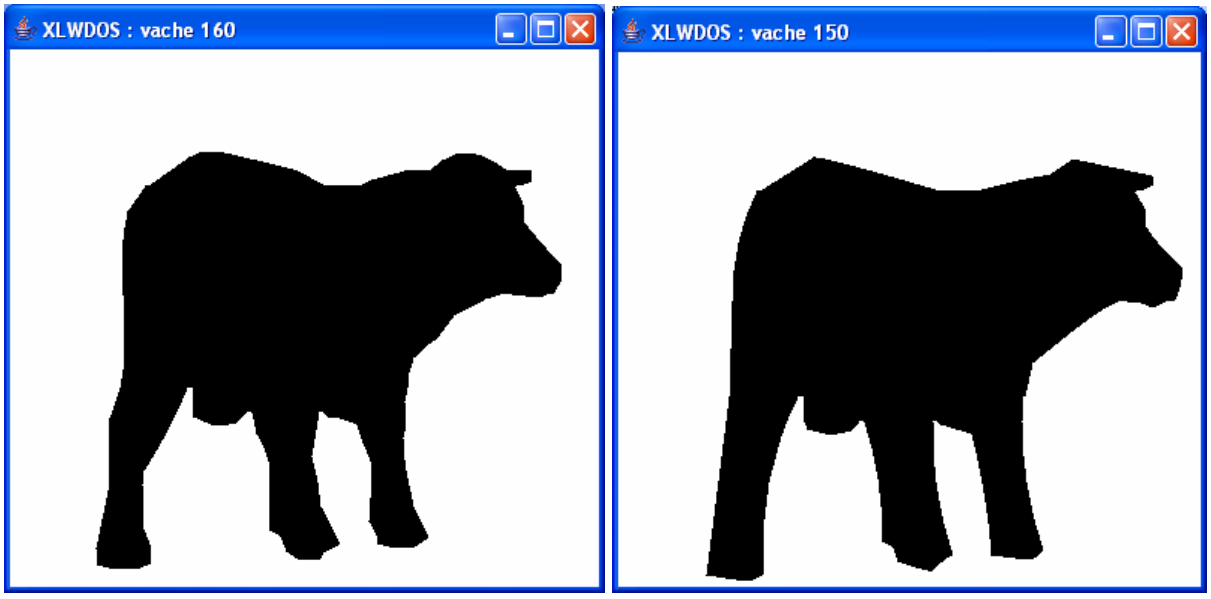

Figure 5.20 : Reconstruction de « vache » pour (α = 160, n = 4) et (α = 150, n = 4)

5.3.5. Interprétation des résultats de la reconstruction

- Les résultats de la reconstruction de silhouettes à partir de leurs descripteurs XLWDOS sont liés à la précision de la description. Plus les points de courbure sont nombreux, plus la qualité de la silhouette reconstruite est bonne.
Le choix des paramètres de précision : l'angle α et la distance n, est très important.
- Pour les silhouettes ayant des frontières avec une petite distance entre les points de courbures (exemple *cheval*), on ne peut pas diminuer la distance n au dessus d'une certaine valeur, à défaut de perdre des courbures importantes. Ce qui déforme la silhouette.
Il existe donc des paramètres de précision minimum à ne pas ignorer.
- Pour les silhouettes polygonales, le descripteur reste inchangé lors de la variation des paramètres de précision.

- Nous avons aussi remarqué que les images de logos industriels dont les frontières de leurs silhouettes sont régulières, conviennent bien à leur codage et décodage avec le descripteur XLWDOS.

5.4. Quantification de la qualité de l'image

Comme nous l'avons expliqué au chapitre précédent, la qualité de reconstruction de silhouettes est mesurée par le taux de perte. Pour cela, nous calculons pour chaque silhouette le nombre de pixels en plus et en moins. Nous calculons ensuite le rapport du nombre de ces pixels sur le nombre total des pixels de la silhouette.

Pour pouvoir calculer le taux de perte, nous avons conçu un outil qui prend en entrée deux images, *nom_image1.gif* et *nom_image2.gif*. *nom_image1* étant l'image originale et *nom_image2* l'image reconstruite, l'application les compare suivant la méthode expliquée dans le chapitre 4.

L'application superpose les deux silhouettes à comparer et affiche le résultat codé de la manière suivante :
- le caractère « O » représente les pixels blancs sur les deux images.
- le signe « + » représente les pixels qui n'existent que sur l'image reconstituée.
- le signe « - » représente les pixels qui n'existent que sur l'image originale.
- le caractère « » représente les pixels noirs sur les deux images.

Le nombre de pixels nécessaire au calcul des Taux de Perte est donné aussi; C'est-à-dire le nombre de pixels dans l'image originale, le nombre de pixels en plus dans l'image reconstituée et le nombre des pixels en moins.

Sachant que les images résultant de la reconstruction ne sont pas matricielles, nous avons du les convertir en bitmap (format GIF) en utilisant le logiciel Adobe Photoshop. Afin de pouvoir les comparer en terme de nombre de pixels, avec les images originales qui sont codées sous le format GIF.

A titre d'exemple, nous illustrons pour la silhouette « img01 » le résultat de superposition des deux silhouettes : initiale et reconstruite (voir figure 5.21).

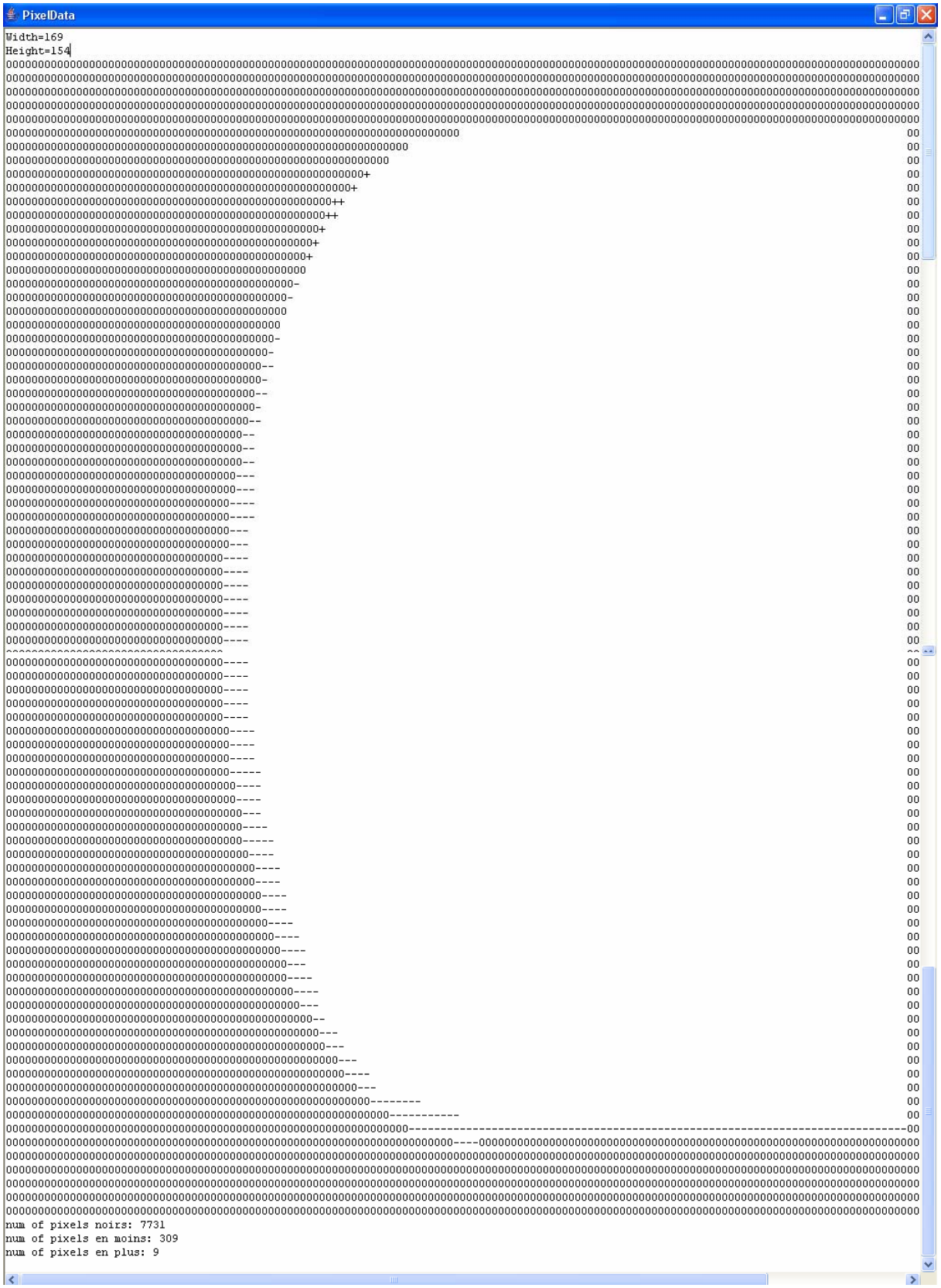

Figure 5.21: résultat de la superposition de deux silhouettes

Nous avons effectué le test de quantification de la qualité sur cinq images parmi celles reconstituées plus haut. Deux silhouettes dessinées : la lettre « T », « img 02 », un logo « hp » et deux images réelles : « poire » et « vache ».
Pour certaines images nous avons réalisé deux visualisations où nous avons fait varier les paramètres α et n, afin de montrer le rôle de la précision du descripteur XLWDOS dans le résultat.

5.4.1. Quantification de la qualité dans le cas d'une image de synthèse

Résultats pour « T » (figure 5.2):
Soit « T_170 » l'image résultat d'un descripteur écrit avec α = 170° et n = 4 pixels.

La comparaison de « T » avec « T_170 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 4860.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 18.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 0.
Ce qui nous donne un TPM = 0,23 % et un TPP = 0 %
Donc, le **TP = 0,23 %**

Résultats pour « img 02 » (figure 5.10):
Soit « img02_170 » l'image résultat d'un descripteur écrit avec α = 170° et n = 4 pixels.

La comparaison de « img02 » avec « img02_170 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 4995.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 116.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 126.
Ce qui nous donne un TPM = 2,32 % et un TPP = 2,52 %
Donc, le **TP = 4,84 %**

5.4.2. Quantification de la qualité dans le cas d'un logo

Résultats pour « hp » (figure 5.11):
Les paramètres de précision utilisés pour décrire le logo sont α = 170° et n = 4 pixels.

1) La forme de gauche :
Nombre de pixels dans l'image originale : 31090.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 468.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 1486.
Ce qui nous donne un TPM = 1,5 % et un TPP = 4,77 %
Donc, le **TP = 6,27 %**

2) La lettre « h » :
Nombre de pixels dans l'image originale : 7908.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 297.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 136.
Ce qui nous donne un TPM = 3,75 % et un TPP = 1,71 %
Donc, le **TP = 5,46 %**

3) La lettre « p » :
Nombre de pixels dans l'image originale : 8533.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 403.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 67.
Ce qui nous donne un TPM = 4,72 % et un TPP = 0,78 %
Donc, le **TP = 5,5 %**

4) La forme de droite :
Nombre de pixels dans l'image originale : 32926.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 775.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 701.
Ce qui nous donne un TPM = 2,35 % et un TPP = 2,12 %
Donc, le **TP = 4,47 %**

5.4.3. Quantification de la qualité dans le cas d'une image réelle

Résultats pour « poire » (figure 5.15) :
Soit « poire_170 » l'image résultat d'un descripteur écrit avec $\alpha = 170°$ et n = 4 pixels. Par contre, « poire_160 » est le résultat d'un descripteur écrit avec $\alpha = 160°$ et n = 4 pixels.

1) La comparaison de « poire » avec « poire_170 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 86454.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 1031.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 2696.
Ce qui nous donne un TPM = 1,19 % et un TPP = 3,11 %
Donc, le **TP = 4,3 %**

2) La comparaison de « poire » avec « poire_160 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 86454.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 2297.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 2057.
Ce qui nous donne un TPM = 2.65 % et un TPP = 2,38 %
Donc, le **TP = 5,03 %**

Résultats pour « vache » (figure 5.19):
Soit « vache_160 » l'image résultat d'un descripteur écrit avec $\alpha = 160°$ et n = 4 pixels. Par contre, « vache_150 » est le résultat d'un descripteur écrit avec $\alpha = 150°$ et n = 4 pixels.

1) La comparaison de « vache » avec « vache_160 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 51368.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 2436.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 1097.
Ce qui nous donne un TPM = 4,74 % et un TPP = 2,13 %
Donc, le **TP = 6,87 %**

2) La comparaison de « vache » avec « vache_150 » nous donne le résultat suivant :
Nombre de pixels dans l'image originale : 51368.
Nombre de pixels en moins dans l'image reconstruite par rapport à l'originale : 2489.
Nombre de pixels en plus dans l'image reconstruite par rapport à l'originale : 1954.
Ce qui nous donne un TPM = 4,84 % et un TPP = 3,8 %
Donc, le **TP = 8,64 %**

5.4.4. Interprétation des résultats de la quantification

Nous pouvons conclure d'après ces résultats que :

- plus les paramètres de précision augmentent, plus le *taux de perte* diminue
- le *taux de perte* est quasiment nul lorsque la silhouette ne contient pas de courbures
- plus le nombre de segments qui constituent l'image est grand, plus le *taux de perte* augmente. Car un segment décalé d'un pixel répercute la déformation sur le reste de la silhouette.

5.5. Espace mémoire occupé

Nous allons prendre quelques images reconstruites plus haut, les enregistrer sous différents formats afin de voir la différence de taille entre les formats qui utilisent les méthodes de compression classique et le format textuel XLWDOS.

5.5.1. Les images dessinées

Nous prenons comme exemple les deux images « T » et « Img02 » pour illustrer l'espace mémoire occupé relativement aux différents formats existants.

formats	« T »	« Img02 »
Gif	**553 octets**	**1,29 ko**
Jpg	**3,07 ko**	**4,03 ko**
Png	**535 octets**	**1,31 ko**
XLWDOS $\alpha = 170°$, n = 4	461 octets, après compression:**313 octets**	2 ko, après compression: **535 octets**

5.5.2. Les logos industriels

Nous avons calculé la somme des tailles des descripteurs des différentes silhouettes constituant un logo et la comparer avec la taille de l'image originale au format vectoriel.

formats	« hp »	« pepsi »
Format vectoriel	22,2 ko	594 ko
XLWDOS	7,20 ko	13,2 ko

5.5.3. Les images réelles

Nous prenons comme exemple les deux images « poire » et « vache » pour illustrer l'espace mémoire occupé relativement aux différents formats existants.

formats		poire	vache
Gif		**2,46 ko**	**2,79 ko**
Jpg		**16,4 ko**	**19,7 ko**
Png		**5,72 ko**	**5,86 ko**
XLWDOS	α = 170° ,n = 4	2,47 ko, après compression:**685 octets**	/
	α = 160° ,n = 4	1,05 ko, après compression:**497 octets**	/
	α = 160° ,n = 5	/	2,62 ko, après compression: **768 octets**
	α = 150° ,n = 5	/	3,79 ko, après compression: **934 octets**

Le format textuel reste le moins gourmand en taille mémoire même devant les formats de compression les plus utilisés.

5.6. Conclusion

Dans ce chapitre nous avons montré que l'opération du codage textuel d'images XLWDOS peut être inversée. Nous avons réalisé des reconstitutions d'images à partir de leurs descripteurs XLWDOS en variant les paramètres de précision. Les résultats diffèrent en qualité de l'image et en taille du fichier XLWDOS.

Concernant les silhouettes dessinées polygonales, les paramètres de précision les plus bas, c'est-à-dire ceux permettant de détecter les points de courbure les plus marquants, suffisent à obtenir une silhouette très proche de l'originale.

Par contre, les résultats sont moins parfaits pour les silhouettes dessinées constituées principalement de courbures.

Pour les silhouettes réelles, il faut augmenter le niveau de précision, donc augmenter la taille du fichier XML, pour obtenir un bon résultat. Pour les paramètres de précision minimum, les frontières des silhouettes sont lissées. Ceci est dû au remplacement d'un ensemble de courbes par une seule.

Nous pouvons conclure que XLWDOS n'est pas plus adapté à des silhouettes particulières plus qu'à d'autres. Il donne de bons résultats d'affichage aussi bien que pour des silhouettes dessinées, des logos ou des silhouettes d'images réelles.

Dans les deux cas, *silhouette dessiné ou réelle*, la taille du fichier contenant le descripteur XLWDOS est plus petite que la taille des fichiers gif, jpg ou même png, avec une qualité de décodage satisfaisante.

Conclusion générale

Dans ce travail, nous avons proposé une nouvelle écriture XML du descripteur de silhouette : XLWDOS.

Ce standard XML nous permet de stocker l'image, ou plus précisément les caractéristiques des segments qui constituent l'image, dans un fichier textuel.

XLWDOS bénéficie des avantages de XML, notamment, sa compatibilité avec la plupart des formats de bases de données et sa capacité à contenir des informations sémantiques.
Mais l'avantage principale d'un tel descripteur reste sa petite taille comparée aux autres formats qui utilisent les méthodes standard de compression d'image.

Nous avons aussi proposé une méthode qui permet de reconstruire et visualiser les images à partir de leurs descripteurs montrant ainsi que le processus de codage XLWDOS peut être inversé.

Les résultats de la visualisation ont été étudiés et comparés aux images originales. Nous en concluons que plus la description s'est faite avec précision, plus l'image reconstruite est de qualité. Nous avons aussi remarqué que les images dont les contours sont réguliers (logos industriels sont un exemple parmi d'autres) répondent mieux à ce nouveau format.

Le gain en taille mémoire fait diminuer considérablement le problème de stockage et de transfert d'images. La taille d'une image peut être réduite d'avantage (de 50% à 20%) en utilisant le codage XLWDOS. Par contre, l'utilisation du processeur est requise afin de le décoder et de visualiser l'image reconstruite.

Un travail est en cours pour le décodage et la visualisation de descripteurs d'images où l'image est considérée comme étant un ensemble de régions (silhouettes). Nous proposons comme perspective à notre travail :

- l'étude de l'apport des différentes méthodes de décomposition de la frontière externe d'une silhouette sur la qualité de l'image reconstruite,
- l'intégration de la couleur de la silhouette dans sa description.

Bibliographie

[1] Alain Michard « *XML Langage et applications* »
EDITION EYROLLES 1999.

[2] Antoine Mircourt « *Le développeur JAVA 2* »
édition Osman Eyrolles Mutimedia 1999.

[3] B. Scassellati, S. Slexopoulos and M. Flickner. *Retrieving Images by 2D Shape A Comparison of Computation Methods with Human Perceptual Judgments*. In SPIE Proc. Storage and Retrieval for Image and Video Databases II, Vol.2185, pp.2-14, 1994.

[4] B. Leibe and B. Schiele, Analyzing appearance and contour based methods for object categorization, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003

[5] C. Boudry. « *La compression d'images* ».
URFIST de Paris/Ecole des Chartes février 2005.

[6] D. S. Zhang and G. Lu. "*Review of Shape Representation and Description Techniques*". Pattern Recognition, 37(1):1-19, 2003.

[7] D. M. Squire and T. M. Caelli. *Invariance Signature: Characterizing Contours by Their Departures from Invariance*. Computer Vision and Image Understanding, 77:284-316, 2000.

[8] Erik T. Ray «*Learning XML*»
First Edition, January 2001

[9] Elliote Rusty Hariold et W. Scott Means *"XML IN A NUTSHELL* Manuel de référence "
EDITION O'REILLY 2001

[10] F. Mokhtarian, A. K. Mackworth, *A theory of multiscale, curvature-based shape representation for planar curves*, IEEE PAMI, Vol 14, N° 8, August 1992

[11] Gabriel Peyré, Ismael Castillo, Charles Dossal, Abdelwaheb Di «*La compression d'images* »
http://www.cmap.polytechnique.fr

[12] G. Granlund. "*Fourier Preprocessing for Hand Print Character Recognition*". IEEE Trans. Computers, 21:195-201, 1972.

[13] G. J. Lu and A. Sajjanhar. "*Region-based Shape Representation and Similarity Measure Suitable for Content-based Image Retrieval*". Multimedia Systems, 7(2):165-174, 1999.

[14] Jean-Claude SOHM « *La compression d'images* »
CERIG / EFPG - 02 décembre 2003.

[15] Irène Charon « *Le Langage Java* »
Hermes Science. Février 2003.

[16] Laurent MYARA- « *La compression »*
Supinfo, Ecole Supérieure d'Informatique-2005

[17] M. Sonka, V. Hlavac, R. Boyle. « *Image Processing, Analysis and Machine Vision"*.
Chapman & Hall Computing, 1993.

[18] Michael Reed Teague. "*Image Analysis Via the General theory of Moments"*. Journal of Optical Society of America, 70(8):920-930, 1980.

[19] Sophia Antipolis - Cours de Traitement d'Images
Université de Nice - 2005-2006

[20] S. Belongie, J. Malik, J. Puzicha, "*Shape matching and object recognition using shape contexts"*, IEEE Transactions on P.A.M.I., Vol. 24, n° 24, 2002

[21] S. Larabi, S. Bouagar, F. M. Trespaderne, E. F. Lopez, LWDOS: *"Language for Writing Descriptors of Outline Shapes"*, In the LNCS proceeding of Scandinavian Conference on Image Analysis, June 29 - July 02, Gotborg, Sweden, 2003

[22] T. Pavlidis. *"Algorithms for Shape Analysis of Contours and Waveforms"*. IEEE Trans. PAMI- 2(4):301-312, 1980.

[23] T. Zaharia, F. Prêteux, « *3D versus 2D/3D Shape Descriptors* » Unité de projets ARTEMIS.

[24] T. Kato. *"Database Architecture for Content-based Image Retrieval"*. In Image Storage and Retrieval Systems, Proc SPIE 1662, pp.112-123, 1992.

[25] T. Zaharia, F. Prêteux, « *3D versus 2D/3D Shape Descriptors :A comparative study* » Groupe des Ecoles des Télécommunications- Unité de projets ARTEMIS.

[26] W. I. Groskey and R. Mehrotra, *"Index-based Object Recognition in Pictorial Data Management"*. Computer Vision, Graphics, and Image Processing, 52:416-436, 1990

[27] W. I. Groskey, P. Neo and R. Mehrotra. *"A Pictorial index mechanism for Model-based matching"*. Data & Knowledge Engineering, 8:309-327, 1992.

[28] le XML en Java.
SUPINFO- Ecole Supérieur d'Informatique de Paris. 21 octobre 2004.

[29] Langage de balisage extensible (XML) 1.0
Recommandation du W3C, 10 février 1998 http://www.w3.org/TR/REC-xml

[30] http://www.megginson.com/SAX/. Site dédié à SAX

[31] http://www.w3.org/TR/REC-png.pdf recommandation png

[32] CCM : Encyclopédie informatique libre