

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
« HOUARI BOUMEDIENE »



Faculté d'Electronique et Informatique
Département d'Informatique

Mémoire

Présenté pour l'obtention du grade de MAGISTER
en informatique

Option : Programmation et systèmes

Par : Mr ABDELAZIZ ALIOUCHE

Thème

**Visualisation d'images à partir de leurs
descriptions textuelles**

Soutenu le 10/07/2007, devant le jury composé de :

Mme A. AISSANI	(Professeur)	USTHB	Présidente de jury
Mr SLIMANE LARABI	(Maître de conférence)	USTHB	Directeur de thèse
Mr A.K. BELKHIR	(Maître de conférence)	USTHB	Examineur
Mme F. SOUAMI	(Maître de conférence)	USTHB	Examinatrice
Mr H. AZOUNE	(Maître de conférence)	USTHB	Examineur

Résumé :

La recherche d'images dans une base de données qui est un thème d'actualité en multimédia et en vision artificielle, dépend fortement de la représentation de l'image.

De nombreux travaux ont été réalisés sur la représentation des formes en se basant sur leur contour externe. Dans ce même axe de recherche, une méthode a été développée au laboratoire LRIA (USTHB) permettant de décrire l'ensemble des régions constituant l'image moyennant un Langage appelé LWDI (*Language for Writing Descriptors of Images*) avec un format XML.

Dans ce mémoire, il s'agit de montrer qu'il est possible de reconstruire l'image à partir de sa description textuelle. Il s'agit ensuite de proposer les possibilités d'utilisation de ce nouveau format suite aux résultats obtenus sur des images réelles et artificielles. Nous proposons donc les différents algorithmes nécessaires pour effectuer cette reconstruction. Les qualités d'images reconstruites sont ensuite quantifiées et commentées.

Mots clés : Description, Silhouette, Région, Partition, Image, Langage.

Abstract:

The image retrieval that is an actual research topic in multimedia and computer vision depends strongly on the image representation.

Many methods proposed for shape representation are based on their external contour. One of these methods is proposed at the LRIA (USTHB) laboratory allowing the description of all regions of the image by means of an XML language LWDI (*Language for Writing Descriptors of Images*)

In our work, we show firstly that is possible to draw the image starting from its LWDI descriptor. We investigate after what are the possibilities of the use of this new image format. We propose thus all algorithms and the results of visualized images and we give the measures of their qualities.

Keywords : Descriptor, Silhouette, Region, Partition, Image, Language.

TABLE DES MATIERES:

<u>TITRES</u>	<u>PAGES</u>
Résumé du mémoire	01
Table des matières	02
Table des figures	07
Introduction générale:	09
Chapitre 1: Codage d'image:	
I. Introduction:	11
1 Images matricielles:	11
2 Images vectorielles:	12
3 Métafichiers:	12
II. Compression d'image:	12
1 Critères de comparaison entre les techniques de compression:	13
2 Algorithme de compression d'images:	13
a) Algorithme de compression sans perte:	13
1. Algorithme RLE:	13
2. Algorithme LZW:	14
3. Le codage Huffman:	14
b) Algorithme de compression avec perte:	15
1. Compression JPEG:	15
2. Compression en Ondelette:	16
3. Compression fractale:	17
III. Formats des fichiers images:	17
1. Formats matriciels:	17
a) Format BMP:	17
b) Format JFIF:	18
c) Format GIF:	20
d) Format PNG:	22
e) Format PCX:	23
f) Format TIFF:	24

2. Formats vectoriels:	25
a) Format SWF:	25
b) Format SVG:	26
c) Format FIF:	28
3. Formats mixtes:	28
a) Format WMF:	28
b) Format PDF:	29
c) Format EPS:	29
d) Format DCS:	30
e) Format PCT:	31
IV. Comparaison:	31
1 Comparaison entre les formats étudiés:	31
2 Formats bitmaps:	33
3 Formats vectoriels:	33
V. Conclusion:	34

Chapitre 2: Description textuelle d'image:

I. Introduction:	35
II. Segmentation d'images:	35
1 Définition:	35
2 Approches de segmentation:	36
3 Méthode de segmentation JSEG:	36
III. Méthode de description d'image:	37
1 Décomposition de l'image en régions:	37
2 Description de silhouette	37
1. Définition:	37
2. Décomposition de silhouette:	37
3. Description de partition:	39
– Description d'un contour élémentaire:	39
– Description de frontière:	39
4. Description de ligne de jonction/disjonction:	40
5. Description d'une partie composée:	40
3 Langage XLWDOS (Description de silhouette):	41

a) Eléments de base du langage:	41
b) Règles d'écriture d'un descripteur LWDOS:	42
c) Ecriture XML d'un descripteur LWDOS	42
d) Exemple de description:	43
4 Description de régions:	44
1. Définition:	44
2. Niveau d'une région interne:	44
3. Description des régions et régions internes en LWDI:	44
a) Eléments de base du langage:	44
b) Règles d'écriture d'un descripteur LWDI:	45
4. Ecriture XML d'un descripteur LWDI d'une région	45
5. Exemple:	46
5 Description de l'image globale:	46
a) Eléments de base du langage:	47
b) Syntaxe du langage LWDI:	47
c) Ecriture XML du descripteur LWDI global	47
d) Exemple:	48
IV. Caractéristiques du format LWDI:	48
1. Caractéristiques:	48
2. Compression du format:	49
3. Comparaison avec d'autres formats:	49
V. Conclusion:	50

Chapitre 3: Reconstruction d'image à partir de son descripteur LWDI:

I. Introduction:	51
II. Unicité de reconstruction:	51
1. Reconstruction d'une Silhouette:	52
2. Reconstruction d'une région:	54
3. Reconstruction de l'image:	55
III. Méthode de reconstruction:	55
1. Schéma général:	55
2. Principes de base et algorithmes:	57

1. Reconstruction de silhouette:	57
a. Reconstruction des contours élémentaires:	57
b. Reconstruction des partitions:	59
c. Reconstruction des lignes de jonction:	61
d. Reconstruction de parties composées:	63
2. Reconstruction de région:	67
a. Positionnement de région:	67
b. Colorisation de région:	68
c. Régions internes:	68
d. Explication:	69
3. Reconstruction de l'image:	70
3. Démonstration:	71
4. Exemple de reconstruction:	74
IV. Etude de performances:	78
1. Efficacité:	78
a. Nombre de contour:	78
b. Nombre de points par région:	79
c. Nombre de région:	79
2. Qualité de l'image reconstruite:	80
1. Introduction	80
2. Méthode proposée :	81
a. Paramètres influant sur la précision:	81
b. Méthode de quantification:	82
a. Quantification de la géométrie	82
b. Quantification de la couleur	82
c. Quantification de la qualité globale	82
V. Conclusion:	84
 Chapitre 4: Expérimentations:	
 I. Introduction:	85
II. Outils nécessaires:	85
1. Segmentation d'image:	85
2. Description de région:	85

3. Description de silhouette:	86
III. Reconstruction d'image par l'application développée:	86
1. Présentation de l'application:	86
a. Structure de données utilisées:	86
b. Vérification du descripteur:	87
c. Caractéristiques de l'application:	87
2. Reconstruction d'image:	88
a. Image de synthèse:	88
1. Reconstruction de Silhouette:	88
a) Silhouette polygonale:	88
b) Silhouette avec courbures:	89
2. Reconstruction d'image:	90
a) Image simple:	90
b) Image complexe:	94
b. Images réelles:	99
a) Logo "A&E":	99
b) Logo "7UP":	105
3. Discussion des résultats:	113
IV. Conclusion:	114
Conclusion générale:	115
Bibliographie:	116

TABLE DES FIGURES:

<u>Figures</u>	<u>Pages</u>
Figure 1.1: Structure d'un fichier JFIF	19
Figure 1.2 : Structure du format GIF	20
Figure 1.3: Structure d'un segment (chunk)	22
Figure 1.4 : Structure d'un dictionnaire	26
Figure 1.5: Fichiers accompagnant le format DCS	30
Figure 2.1 : Segmentation d'image	37
Figure 2.2: Rectangle d'aire minimum	38
Figure 2.3 : Décomposition de silhouette	38
Figure 2.4 : Contour élémentaire	39
Figure 2.5 : Notion de partie composée	40
Figure 2.6 : Exemple de description de silhouette	43
Figure 2.7 : Exemple description de région	46
Figure 2.8 : Image de comparaison	49
Figure 3.1: Silhouette	52
Figure 3.2: Décomposition de la silhouette	52
Figure 3.3 : Unicité de reconstruction d'un contour élémentaire	53
Figure 3.4: Structure d'une région	54
Figure 3.5 : Décomposition d'une région	54
Figure 3.6 : Structure d'une image	55
Figure 3.7: Schéma général	55
Figure 3.8 : Schéma général du traitement	56
Figure 3.9 : Eléments d'un Contour	57
Figure 3.10 : Calcul du centre d'une courbe	59
Figure 3.11 : Sens de reconstruction d'une partition	59
Figure 3.12: Sens et ordre de reconstruction des segments	61
Figure 3.13 : Reconstruction d'une partie composée	63
Figure 3.14: Exemple de reconstruction d'une partie composée	66
Figure 3.15: Structure d'une région	67
Figure 3.16 : Positionnement d'une région	67
Figure 3.17 : Reconstruction d'image	70
Figure 3.18: Schéma d'exécution (Cas1)	71
Figure 3.19: Schéma d'exécution	73
Figure 3.20: Schéma d'exécution (Cas3)	73
Figure 3.21: Schéma d'exécution (Cas général)	73
Figure 3.22 : Exemple de reconstruction	74
Figure 3.23: Schéma d'exécution (exemple)	76
Figure 3.24 : Point de courbure selon l'algorithme de Chetverikov	78
Figure 3.25: Image d'exemple 'originale'	83
Figure 3.26: Image d'exemple 'reconstruite'	83
Figure 4.1 : Silhouette polygonale originale	89
Figure 4.2: Reconstruction de silhouette l	90
Figure 4.3 : Image "Bateau" originale	90
Figure 4.4: Reconstruction de l'image "Bateau"	90

Figure 4.5: Image "Fruits" originale	91
Figure 4.6: Reconstruction de l'image fruit	92
Figure 4.7: Image complexe	95
Figure 4.8: Reconstruction d'image complexe	96
Figure 4.9: LOGO "A&E" original	100
Figure 4.10: LOGO "A&E" reconstruit	101
Figure 4.11: LOGO "7UP" original	106
Figure 4.12: LOGO "7UP" reconstruit	107

INTRODUCTION

Le besoin en diverses informations est devenu pour l'homme d'aujourd'hui une nécessité de vie, tout comme son besoin à l'air, à l'eau et à la nourriture. Parmi ces quantités diverses et gigantesques d'informations qui circulent quotidiennement à travers le monde, celle qui est visuelle (image et vidéo) à certainement un effet beaucoup plus attirant que celle qui est écrites.

«Une belle image vaut mieux qu'un long discours », cette expression explique bien l'importance de l'image notamment dans quelques domaines où le besoin en diverses informations de qualité est immense, et s'accroît d'un jour à l'autre.

Sur le Web l'image joue un rôle incontournable et s'impose comme un support et une source d'informations privilégiés, d'une part en raison du charme qu'elle donne au pages Web, et d'autre part de sa capacité à aider le visiteur à mieux identifier le contenu d'un site en évitant d'avoir recours à des textes trop explicatifs et trop longs dans certains cas.

Actuellement malgré l'évolution rapide des équipements électroniques qui permettent la manipulation d'image numérique, celle-ci reste souvent très lourde surtout en se qui concerne sa transmission sur réseaux.

De nombreuses techniques de compression ont été développées afin de réduire le volume de l'image numérique, certaines sont très complexes et exploitent de multitudes fonctions mathématiques, ce qui requiert un temps machine énorme pendant les opérations de compression et décompression, mais vu la rapidité des machines actuelles ceci reste souvent acceptable vu le gain obtenu en espace mémoire.

Ces techniques de compression ont donné naissance à une multitude de formats d'image répartis en trois grandes catégories: la première catégorie celle des formats bitmap, se contente de représenter l'image sous forme d'une matrice et applique les différentes techniques de compression pour réduire le volume de l'image, la deuxième catégorie celle des formats vectoriels se base sur la représentation de l'image sous formes de vecteurs de données exploités par des fonctions mathématiques, et la troisième catégorie celle des métafichiers qui combine les deux techniques en essayant de se bénéficier des avantages de chacune d'elles.

Actuellement la recherche d'image sur le net ou dans une base de données multimédias en se basant sur les critères de recherche classiques (nom, date, volume, type) est quasiment navrant, et devient de plus en plus difficile du fait de l'augmentation énorme de l'information potentielle intéressante.

Des techniques de recherche d'image basées sur quelques caractéristiques comme la couleur, les textures ou les formes d'objets existent déjà mais restent souvent très limitées. Le besoin à des nouvelles techniques permettant la recherche sémantique d'image s'est devenu ultimement demandé.

Dans cet axe de recherche, une méthode de description d'image a été développée au laboratoire L.R.I.A de l'USTHB. Son principe de base consiste à décomposer l'image en régions représentée par une structure de graphe [LAR07].

Cette représentation est ensuite traduite moyennant un langage appelé **LWDI** (Language for **W**riting **D**escriptors of **I**mages) développé au même laboratoire, en une description textuelle suivant le format XML.

Chaque région est décrite par sa position dans l'image, sa couleur et la géométrie de son contour externe. La description du contour externe est basée sur un langage **LWDOS** (Language for **W**riting **D**escriptors of **O**utlines **S**hapes) qui décrit avec un format XML la géométrie des différentes parties constituant la silhouette (partitions, lignes de jonction et de disjonction) [LAR 03].

Un descripteur LWDI non seulement il offre de meilleures conditions de stockage et de transfert, du fait qu'il occupe peu d'espace mémoire, mais il offre aussi une facilité d'indexation, ce qui lui rend très utile dans le domaine de recherche d'image.

Il s'agit de trouver les possibilités d'application de ce nouveau format d'images dans le domaine des bases de données images et dans le monde du web. Pour cela, nous proposons dans ce travail une méthode pour le décodage du descripteur LWDI, et la reconstruction de l'image à partir de celui-ci. Il s'agit ensuite d'évaluer les performances de nouveau format (qualité d'image reconstruite, temps de reconstruction).

Dans ce mémoire nous avons réparti notre travail en quatre chapitres:

Dans le premier chapitre nous présentons l'étude réalisée sur les formats d'image les plus utilisés, ainsi que les techniques de compression exploitées dans ce domaine. Ceci nous permet de mettre en évidence les différents avantages et inconvénients caractérisant chaque catégorie de formats d'image.

Dans le deuxième chapitre nous décrivons la méthode de description textuelle d'image et le langage LWDI, tout en donnant des exemples de description d'image.

Dans le troisième chapitre nous proposons une méthode pour la reconstruction d'image à partir de son descripteur LWDI. Nous décrivons aussi les différents algorithmes nécessaires pour la reconstruction de chaque élément de l'image. Nous proposons ensuite une méthode de quantification du résultat qui permet d'évaluer la qualité de l'image obtenue par rapport à l'image originale.

Dans le quatrième chapitre nous donnons les résultats obtenus pour la reconstruction d'images artificielles et réelles.

Nous terminons notre travail par une conclusion générale dans laquelle nous donnons les perspectives éventuelles de celui-ci.

CODAGE D'IMAGES

I- Introduction :

Le mot codage signifie selon le dictionnaire « l'action d'appliquer un code (règles ou signes) pour transformer un message ou des données en vue de leur transmission ou de leur traitement ».

Dans notre cas -étant donné qu'une image est un ensemble de points- on désigne par codage d'image la manière dans laquelle les différents points constituant celle-ci sont représentés. C'est ce qu'on appelle « *Format d'image* ».

Une multitude de formats d'images est utilisée aujourd'hui, ceci est lié d'une part à la concurrence entre les fabricants des logiciels et la volonté de chacun d'eux à imposer son propre format, et de l'autre part aux différentes caractéristiques d'une image (dimensions, couleurs, fixe/dynamique, compression etc..) [PIE 99].

Certains formats sont plus utilisés que d'autres, ceci est dû essentiellement à la qualité et le volume de l'image.

Une image est une suite de points adjacents appelés "*Pixels*". La manière la plus facile pour représenter une image est de la considérer comme étant une matrice de pixels (Image Matricielle ou Bitmap) dont chacun est représenté par sa couleur et sa position dans l'image.

L'image résultante dans ce cas est souvent très volumineuse ce qui rend l'appel à une technique de compression indispensable pour pallier ce problème. Une autre manière de représentation d'image consiste à faire une description de l'image en faisant appel à des fonctions mathématiques (géométriques), ce genre d'image est appelé "Image Vectorielle".

I-1- Images matricielles:

L'image est considérée comme étant une matrice composée d'une suite de lignes, qui sont elles-mêmes composées de suites de points appelés pixels.

Ce système de codage consiste donc à décomposer la représentation de l'image, en un certain nombre de points élémentaires, chaque point est caractérisé par ses coordonnées spatiales et sa couleur choisie parmi la palette de couleurs utilisée [HEN 03] [SED 02]. La qualité et le volume de l'information codée et transmise dépendront donc des deux éléments suivants:

- a. **L'information donnée par un pixel :** le nombre de couleurs (NC) que peut contenir une image dépend directement du nombre de bits (n) utilisés pour la représentation d'un pixel ($NC = 2^n$).
- b. **La densité de ces pixels (résolution de l'image) :** des recherches scientifiques ont montré que l'œil humain est capable de distinguer deux points séparés de 3 mm à une distance de 10m, ainsi pour avoir une image de bonne qualité, la résolution de celle-ci doit être au moins de 3 pixels par mm.

Ces deux éléments montrent la forte relation entre la qualité et le volume de l'image.

Exemple :

Soit une image de 600x400 pixels codée sur 24 bits.

Son volume théorique est égal à $600 \times 400 \times 3$ octets = 703,125 ko.

Cet exemple montre combien ce type de formats sont volumineux ce qui rend l'application d'une technique de compression très indispensable, pour gagner de l'espace mémoire dans l'archivage et faciliter leur transmission sur réseaux [BUR 01].

I-2- Images vectorielles :

Les formats vectoriels se contentent de faire une description géométrique de l'image; le document numérisé prend donc la forme d'une suite de formules mathématiques décrivant les formes élémentaires constituant l'image (carrés, rectangles, ellipses, cercles, courbes, etc.). Chaque forme élémentaire constitue un objet et se voit assigné un certain nombre d'attributs tels que la couleur, la transparence, l'épaisseur du trait, le type de trait etc.

Le stockage d'une image du type vectoriel est donc très différent de celui d'une image de type bitmap : il consiste en la mémorisation de la représentation des coordonnées des points caractéristiques des formes qui constituent l'image. Il s'agit dès lors d'une représentation relative que l'on peut qualifier de "symbolique".

I-3- Métafichiers :

Pour bénéficier à la fois des avantages des deux types d'images cités ci-dessus un nouveau type est apparu, ce type permet d'intégrer à la fois des images bitmaps avec des objets vectoriels.

Les formats les plus connus sont EPS, Macintosh PICT, RTF et WMF. Le format WMF est une bonne illustration de l'organisation des données au sein du fichier.

II- Compression d'image :

Le coût et les limites technologiques actuelles nécessitent de diminuer le volume d'image que se soit pour le stockage ou pour la transmission sur réseaux. Beaucoup de techniques de compression de données ont été développées depuis les années soixante, la majorité sont indépendantes des formats de fichiers spécifiques, et peuvent être donc appliquées sur n'importe quel type de fichier [AUJ 04] [WAY 99].

La compression des fichiers images varie selon le type d'image compressée : fichier bitmap, fichier vectoriel ou métafichier. Dans le cas des fichiers bitmap, l'entête, la table des couleurs ainsi que les données autres que l'image, doivent être laissés sans compression pour permettre de repérer la zone contenant les données. Généralement la compression de ce type de fichiers donne un gain considérable en volume [YUN 99].

Les fichiers vectoriels n'incluent généralement pas de compression. En effet, les données étant sous forme de vecteurs mathématiques, elles sont déjà une forme compacte de l'image. De plus, les fichiers de vecteurs sont relativement longs à lire, ajouter une compression ne ferait que rendre la tâche plus difficile [MIN 00].

Les métafichiers utilisent, quand ils comportent des images non vectorielles, les mêmes techniques de compression que les fichiers bitmaps [BAR 06].

II-1- Critères de comparaison entre les techniques de compression :

Les techniques de compression peuvent être comparées selon les critères suivants :

- a) **Efficacité (taux de compression) :** C'est le rapport entre la taille du fichier compressé et sa taille initiale. Pour une meilleure utilisation des algorithmes, il faut comprendre que la plupart d'entre eux sont plus ou moins efficaces dans un type d'image que dans un autre [NEL 93].

$$\sigma = \frac{\text{Nombre de bits dans l'image originale}}{\text{Nombre de bits dans l'image compressée}}$$

- b) **Qualité de compression :** Certains algorithmes permettent une compression avec une perte d'un certain pourcentage de la qualité d'image par contre d'autres permettent une compression sans perte.
- c) **Vitesse de compression/décompression :** Certains algorithmes permettent une compression ou décompression de l'image avec une vitesse plus ou moins rapide que d'autres.
- d) **Accessibilité :** Certains algorithmes de compression sont privés, leur utilisation se fait uniquement sous licence, par contre que d'autres sont libres de droits et peuvent être utilisés librement. Ce critère peut avoir son importance tant en termes de coût de création que de pérennité à long terme [BRO 03] [NEL 93].

II-2- Algorithmes de compression d'images :

Le choix d'une technique de compression à appliquer sur une image dépend du domaine d'utilisation de celle-ci. Par exemple pour les images destinées au Web on préfère utiliser des images de petites tailles même si elles sont d'une qualité réduite que d'utiliser des images en haute qualité mais qui prennent beaucoup de temps dans leur téléchargement, par contre pour l'archivage d'image on préfère avoir des images de bonne qualité.

Les algorithmes de compression d'images peuvent être répartis en deux classes selon les deux cas de compression cités ci-dessus [NEL 93] [NGA 99] :

a) Algorithmes de compression sans perte :

Ces méthodes sont basées généralement sur la recherche et le codage des données redondantes. Ce type de compression peut s'appliquer à n'importe quel type de données.

1. Algorithme RLE (Run Length Encoding) :

C'est la méthode la plus simple et la plus utilisée. Son principe de base consiste à rechercher des données redondantes (pixels dans le cas des images) et en codant la longueur et la valeur.

Ainsi à chaque répétition d'un pixel plus d'un nombre n précisé par l'utilisateur, cette suite de pixels est remplacée par un caractère spécial indiquant la

compression suivi par le nombre de répétitions du pixel et en fin sa valeur [NEL 93] [BRO 03].

❖ **Caractéristiques :**

- Algorithme de compression ou de décompression très simple à implémenter.
- Taux de compression relativement faible par rapport à d'autres algorithmes.
- Obtient des meilleurs résultats avec des images contenant des zones importantes de couleur contiguë (images monochromes). La compression des images en couleurs complexes (photos) ne donne pas des bons résultats [MIN 00].

2. La compression LZW (Lempel Ziv Welch) :

Cet algorithme est une amélioration faite par *Terry Welch* de la société *Unisys* en 1984, sur l'algorithme LZ77 et LZ78 développés par *Abraham Lempel* et *Jakob Ziv* en 1977 et 1978 [BER 01].

Il est basé sur un dictionnaire (bibliothèque) construit au fur et à mesure de la lecture du fichier à coder. Les chaînes de caractères sont placées une par une dans la bibliothèque. Lorsqu'une chaîne est déjà présente dans la bibliothèque, son code de fréquence d'utilisation est incrémenté. Les chaînes de caractères ayant des codes de fréquences élevés sont remplacées par un " mot " ayant un nombre de caractères le plus petit possible et le code de correspondance est inscrit dans la bibliothèque. On obtient ainsi l'information encodée et sa bibliothèque [BER 01].

❖ **Caractéristiques :**

- Cet algorithme est breveté par la société Unisys, il a été utilisé dans les formats TIFF et GIF, par contre l'algorithme LZ77 est libre de droit et a été utilisé dans le format PNG.
- Il s'applique très bien sur les images de faibles profondeurs (nombre réduit de couleurs différentes) puisque les motifs différents doivent être relativement faibles pour être répétés.
- Il est l'un des plus répandus algorithmes, et est très rapide aussi bien en compression qu'en décompression [NEL 93] [BOU 04].

3. Le codage de Huffman :

Cet algorithme est développé en 1952 par David Huffman, il est l'un des algorithmes les plus anciens, son codage est basé sur la fréquence d'apparition d'un caractère : plus le caractère apparaît souvent plus son code sera court et vice-versa. Pour permettre un décodage unique les codes attribués aux différents caractères doivent être préfixés, c'est-à-dire qu'aucun caractère n'est un préfixe d'un autre. C'est pourquoi on appelle aussi ce codage un *VLC préfixé (Variable Length Code, code à taille variable)*.

Les codes des caractères sont attribués de la manière suivante [NEL 93] :

- 1) Chercher la fréquence d'apparition de chaque caractère.
- 2) Trier les caractères par ordre décroissant de fréquence (poids).
- 3) Construire l'arbre binaire comme suit :
 - i) Relier deux à deux les caractères de fréquences les plus basses et affecter à ce nœud la somme des fréquences des caractères.

- ii) Répéter l'opération jusqu'à ce que l'arbre relie toutes les lettres.
- iii) L'arbre étant construit, on met un 1 sur la branche à droite du nœud et un 0 sur celle de gauche.
- 4) Parcourir l'arbre de la racine vers chacune des feuilles pour tirer le code de chaque caractère.

❖ **Caractéristiques :**

- Cet algorithme permet d'avoir un taux de compression très élevé (50% en moyenne) et un temps de compression assez rapide.
- La bibliothèque doit être transmise avec le fichier.
- Il est très sensible: la perte d'un bit entraîne une altération de toutes les données qui suivent lors de la décompression [YUN 99].

b) Algorithmes de compression avec perte :

La compression avec pertes (*lossy compression*), se permet d'éliminer quelques informations pour avoir le meilleur taux de compression possible, tout en gardant un résultat qui soit le plus proche possible des données originales. C'est le cas par exemple de certaines compressions d'images ou de sons [AUJ 04][NIC 03].

Etant donné que ce type de compression supprime des informations contenues dans les données à compresser, ces méthodes donc ne peuvent être appliquées que sur les données informatiques volumineuses dont leur qualité se limite à la perception de l'œil humain comme dans le cas des images ou des séquences vidéo [NIC 03]. Son application sur des fichiers exécutables, ou sur des fichiers textes conduits généralement à la perte de ces fichiers [WAY 99] [NEL 93].

Sachant que l'œil humain ne peut apercevoir qu'un nombre restreint de couleurs, et par conséquent certaines nuances ne peuvent être aperçus. Ainsi un algorithme qui sait tirer profit de ces caractéristiques peut offrir d'intéressants taux de compressions [WAY 99].

Ce genre de compression est basé principalement sur les algorithmes suivants :

- La compression JPEG.
- La compression par ondelettes.
- La compression fractale.

1. Compression JPEG :

La compression JPEG (*Joint Photographie Experts Group*) est une des meilleures techniques de compressions d'images fixes actuellement sur le marché, et certainement la plus utilisée [KIV 96]. Elle permet d'obtenir des meilleurs taux de compression avec des images de 24 bits (vraie couleur) complexes [ACH 05].

Elle procède en appliquant une technique mathématique dite DCT (*Discrete Cosine Transform*) pour éliminer les données d'image imperceptibles à l'œil humain nu. Ensuite le codage *Huffman* sera appliqué pour accroître la compression [BOU 04] [WHI 99] [JER 01].

Cet algorithme permet de procurer un bon compromis entre les deux facteurs formants le tandem inséparable: taux de compression et qualité d'image, en laissant à l'utilisateur de définir le niveau de compression, à l'aide d'un paramètre de qualité abstrait [GUI 96] [PEN 92].

Trois extensions sont utilisées pour l'algorithme de base :

- Le JPEG progressif : permet un décodage et un affichage progressif de l'image.
- Le JPEG arithmétique : permet d'avoir des taux de compression très élevés.
- Le JPEG sans perte : emploie un algorithme différent de celui du JPEG de base.

Cette compression est utilisée dans les formats JFIF, SPIFF et TIFF [PEN 92].

2. Compression en ondelettes :

La transformation en ondelettes est utilisée en mathématique depuis les années trente, sous le nom de décomposition de *Littlewood-Paley* [BOU 04].

Cette théorie qui est en réalité une extension de l'analyse de Fourier a été utilisée dans les années 80 pour la décomposition d'un signal continu en une série de nombres décrivant des courbes qui s'additionnent pour reconstruire le signal [YUN 99][DAV 01].

La compression par ondelettes est basée sur la division de la résolution de l'image en codant la perte de l'information découlant de cette division [WHI 99] [LEC 06].

Ainsi on procède comme suit :

1. On part d'une image A.
2. On divise par 2 la définition horizontale de l'image en moyennant les pixels deux à deux suivant l'axe horizontal, on obtient une image B.
3. Pour chaque pixel, on calcule l'erreur entre l'image originale et l'image B, on obtient l'image C.
4. Pour B et C, on moyenne les pixels deux à deux mais cette fois suivant l'axe vertical, on obtient D et E.
5. Pour B comme pour C, on répète l'étape 3 et on obtient F et G.

L'image D obtenue est une image dont la résolution est divisée par 2 dans les deux sens. Les images E, F et G donnent les erreurs entre A et D. Ainsi la transformation doit être répétée sur l'image D tant de fois jusqu'à atteindre le niveau voulu, et ce en éliminant les images "d'erreurs" inférieures à un certain niveau choisi [ACH 05] [WIT 99].

La dernière étape consiste à utiliser une compression non destructive pour le reste. Pour la décompression il suffit de prendre le même processus à l'envers.

❖ Caractéristiques :

- Elle permet de prévoir le taux de compression contrairement au JPEG.
- Elle n'entraîne pas d'effet de mosaïque.
- L'algorithme est plus simple et plus souple que JPEG et donc plus rapide.
- Il est possible d'avoir des images très compactes.
- Une image compressée par les ondelettes peut être décompressée de deux manières différentes : en fixant sa résolution et augmentant sa taille progressivement ou l'inverse [LEC 06] [MAR 03].

❖ L'algorithme JPEG 2000

Tandis que l'algorithme JPEG est basé sur la transformation *DCT*, ce nouvel algorithme est basé sur la compression par ondelettes pour atteindre un taux de compression plus élevé avec une réduction plus faible de la qualité d'image. Il est destiné à remplacer l'algorithme JPEG. Il permet des compressions avec et sans perte. Cet algorithme est utilisé dans le format JP2 [ACH 05][DAV 01].

3. Compression fractale :

La compression fractale se base sur les travaux de **Benoît Mandelbort**, qui avait suggéré que les images du monde réel peuvent être décrites par des fractales. Une méthode pour trouver les équations fractales qui décrivent une image quelconque a été inventée dans les années 80 par le mathématicien **Michael Barnsley** [BOU 04].

L'application des fractales dans le traitement d'image permet d'améliorer la segmentation de textures et l'extraction du contour [BOU 04].

La transformation fractale consiste à décomposer l'image en un immense puzzle avec des pièces de formes quelconques et à rechercher les pièces qui sont similaires entre elles à une transformation affine après (translations, rotations, symétries et agrandissements ou réductions). Ainsi pour coder l'image, il suffit de décrire les pièces originales et les transformations utilisées [BOU 04] [LEC 06].

❖ Caractéristiques :

- Le temps de compression peut être très élevé (elle demande beaucoup de calculs) par contre la décompression est quasi immédiate. De telles caractéristiques rendent cette technique inutilisable pour des transmissions instantanées. Cependant, pour la transmission d'images sur Internet, la transmission de films stockés dans des banques de données, ou pour le stockage d'information sur CD ROM, cette compression est idéale.
- En forçant le taux de compression, l'image devient floue.
- La taille de l'image n'a pas d'influence sur la taille du fichier [LEC 06] [WAY 99].

III- Formats des fichiers images:

Plusieurs formats contribuent aujourd'hui à l'utilisation des images dans les différents domaines. Certains formats sont plus utilisés que d'autres du fait de leurs caractéristiques différentes, qui sont liées souvent aux types d'images codées (bitmap ou vectorielle) et à la méthode utilisée dans la compression de celles-ci.

III-1- Formats matriciels:

a. Format BMP (*Bitmap Picture*):

Le format BMP est destiné au stockage d'images fixes, et est supporté seulement par Windows et OS/2. L'image est stockée sous forme d'une matrice, dont les éléments de cette matrice sont les différents points (pixels) constituant l'image. Chaque pixel est qualifié par une couleur qui est codée sur 1, 4, 8, 16 ou 24 bits [BUR 01] [JOH 99] [ABA 01].

1. Structure du fichier :

Un fichier BMP est constitué de 3 ou 4 parties comme indiqué ci-dessous:

- 1. L'entête du fichier :** codé sur 14 octets, et contient les informations concernant le fichier (type, taille, adresse du 1^{er} pixel).
- 2. L'entête du bitmap :** codé sur 40 octets, contient les informations concernant l'image (la taille de l'entête, la largeur et la longueur de l'image ainsi que sa taille, le nombre de bits par pixel, le nombre de couleurs, le type de compression...)
- 3. La palette de couleurs :** cette partie est optionnelle, si les couleurs sont codées sur 24 bits cette partie n'existe pas, sinon elle contient la palette de couleurs utilisées.
- 4. Les données de l'image (*Raster*) :** cette partie contient les pixels de l'image, ils sont codés par lignes horizontales, en commençant par la dernière ligne (la ligne en bas de l'image) et en allant de gauche vers la droite [JOH 99].

2. Compression des fichiers BMP :

Les fichiers BMP sont rarement compressés, ce genre de fichier a comme objectif principal la qualité d'image, et la facilité dans la lecture, le codage, l'affichage et le traitement de l'image [BRE 03].

Dans le cas des fichiers compressés, l'algorithme utilisé est RLE qui permet une compression sans perte, et qui doit être appliquée seulement sur les données de l'image, pas sur l'entête. Cependant dans le cas des images réelles cette compression risque de donner des images plus grosses [JOH 99].

3. Caractéristiques et utilisation :

Ce format a pour principal avantage la qualité des images fournies, ainsi que la rapidité dans la lecture et l'affichage de l'image, en plus il est très adapté au traitement d'images. Malheureusement le fait de ne pas compresser le fichier donne des fichiers de grandes tailles, ce qui rend leur utilisation très difficile sur internet. La meilleure solution est d'utiliser ce format pour stocker des images de référence, mais pour l'utilisation sur Internet ces images doivent être converties vers d'autres formats [JOH 99] [GUI 96].

b. Format JFIF (JPEG) :

Le format de fichier JFIF (*JPEG File Interchange Format*) souvent appelé par le nom du groupe fondateur de la compression *JPEG* est l'implémentation de base de la compression avec perte *JPEG*, qui est la plus utilisée actuellement par les applications manipulant des images fixes [LUC 00][HOW 01]. Ce format a été créé par *C-Cube Microsystems*, et est reconnu par les extensions JFIF, JFI, JPG, JPEG [MAR 03].

En effet ce format est capable de stocker des images allant de 2 jusqu'à plus de 16 millions de couleurs, et d'une taille maximale de 65536x65536 points [ACH 05].

1. Structure du fichier JFIF :

Un fichier de format JFIF est structuré de la manière suivante :

Marqueur SOI (<i>Start Of Image</i>)
Marqueur APP0 ('JFIF' +00)
Marqueur APP1 ('UFRGS' compresseur JPEG)
Marqueur DQT (<i>Define Quantization Tables</i>)
Table de quantification
Marqueur SOF (<i>Start Of Frame</i>)
Marqueur DHT (<i>Define Huffman Tables</i>)
Tables Huffman
Marqueur SOS (<i>Start Of Scan</i>)
Données compressées
Marqueur EOI (<i>End Of Image</i>)

Figure 1.1: Structure d'un fichier JFIF

Un fichier JFIF doit contenir en plus des données de l'image, la signature du fichier, les informations concernant le logiciel de compression, la table de quantification et la table *Huffman* utilisées [DAV 01].

Ces données n'ont pas de tailles prédéfinies, c'est pour cette raison qu'il doit utiliser des marqueurs pour séparer chaque type de données d'un autre. Ces marqueurs sont codés sur deux octets et commencent toujours par la valeur 'FF' en hexadécimal.

Les données de l'image viennent toujours après le marqueur SOS, et sont codées dans un seul bloc de haut en bas. C'est uniquement cette partie du fichier qui est compressée [LUC 00] [EHA 92].

2. Compression du format JFIF :

Comme son nom l'indique ce format est basé sur la compression JPEG, ce qui permet d'avoir des taux de compression très élevés, et donne des fichiers de petites tailles. Généralement le taux de compression peut être choisi par l'utilisateur; cependant l'utilisation d'un taux de compression très élevés entraîne une déformation de l'image initiale perceptible par l'œil nu [DAV 01].

Le meilleur taux de compression à utiliser est aux environs de 50% pour faire un compromis entre qualité et taille de l'image [DAV 01].

3. Caractéristiques et applications :

Le format JFIF est l'implémentation de base de la compression JPEG, ce qui lui permet d'hériter tous les avantages et inconvénients de celle-ci:

- Ce format est exploitable par différentes applications et sur différentes plateformes (PC, MAC, UNIX Workstation).
- Le rapport qualité/poids de l'image obtenu est vraiment extraordinaire, surtout pour les images en couleurs réelles. Cet avantage en a fait le format de prédilection de beaucoup d'applications et des appareils photo, souvent même, le seul proposé.

- Ce format peut exploiter la compression JPEG progressive ce qui permet un affichage au fur et à mesure que les données arrivent sur Internet, sans avoir besoin d'attendre l'arrivée de toutes les données de l'image. Ceci le rend un des deux formats les plus utilisés sur Internet (avec le format GIF).
- Ce format est beaucoup plus destiné à afficher des images vues par l'œil humain que d'être destiné au traitement par la machine du fait qu'il est basé sur une compression avec perte.
- La sauvegarde des dessins au trait ou des images contenant des grandes parties de couleurs contiguës dans ce format entraîne l'effet des artefacts, et donne souvent des images médiocres [LUC 00].

c. Format GIF :

Le format GIF (*Graphic Interchange Format*) est créé par la société *CompuserveIncorporated* qui appartient à *UNISYS* [ENC 05], et a connu deux versions, la première version *GIF87a* (sortie en 1987) constitue le standard de ce format [COM 87], la deuxième version *GIF89a* (sortie en 1989) a donné une bonne amélioration au format standard notamment la possibilité de transparence de couleur, l'animation d'images, et l'effet d'entrelacement [NGA 99].

Ce format permet de coder des images animées de 256 couleurs, et de taille maximale de 65536 x 65536 points, tout en gardant un taux qualité/poids d'image très acceptable, ces caractéristiques ont permis à celui-ci d'être un des deux standards de l'Internet [JOH 99] [SED 02].

1. Structure d'un fichier GIF :

Un fichier GIF est constitué de la manière suivante [JOH 99] :

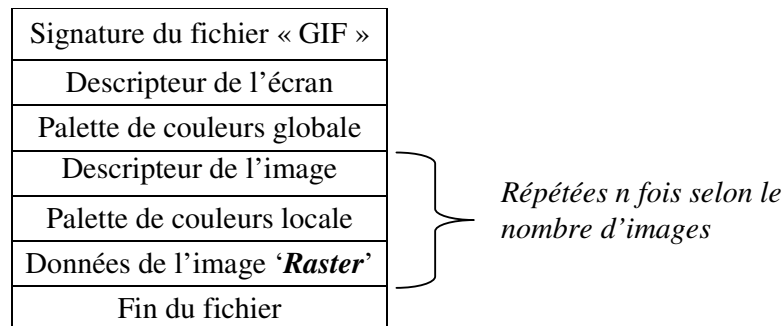


Figure 1.2 : Structure du format GIF

1. **Signature du fichier :** Ce champ est codé sur 6 octets et contient quelques informations qui servent à l'identification du fichier et de sa version.
2. **Descripteur de l'écran :** Cette partie sert à la description de l'écran logique (le rectangle dans lequel l'image peut se déplacer), et la palette de couleurs globale.
3. **Palette de couleurs globale :** Cette palette est optionnelle, le nombre d'entrées dans la palette de couleurs est égal à 2^n où *n* est le nombre de bits par pixel indiqué dans le descripteur de l'écran, et dont chaque entrée est formée de trois octets représentant l'intensité des trois couleurs RVB. En

cas d'absence de cette palette, une palette de couleurs par défaut est créée lors de l'affichage de l'image.

4. **Descripteur d'image :** Ce descripteur (commence toujours par le code '2C' en hexadécimal) permet de décrire l'emplacement d'image actuelle, et celui de l'image suivante par rapport aux dimensions de l'écran logique défini précédemment. Ce descripteur indique aussi l'existence ou non d'une palette de couleur locale et si l'affichage est entrelacé ou non.
5. **La palette de couleurs locale :** Cette palette est optionnelle, en cas de son absence les couleurs utilisées seront celles de la palette globale.
6. **Les données de l'image :** Cette partie contient les valeurs des pixels en allant de gauche à droite, et de haut en bas en séquentiel dans le cas des images séquentielles. Dans le cas des images entrelacées l'affichage se fait en quatre étapes :
 - 1^{ère} passe : affichage de chaque huitième ligne en commençant à partir de la première ligne en haut.
 - 2^{ème} passe : affichage de chaque huitième ligne en commençant à partir de la cinquième ligne en haut.
 - 3^{ème} passe : affichage de chaque quatrième ligne en commençant à partir de la troisième ligne en haut.
 - 4^{ème} passe : affichage du ce qui reste de l'image.

Ces données sont généralement codées selon l'algorithme de compression LZW. Ces trois dernières parties doivent être répétées autant de fois que le nombre d'images élémentaires dans l'image animée.

7. **Fin du fichier :** Cette partie est constituée d'un seul octet, contenant toujours la valeur '3B' en hexadécimal qui correspond au point-virgule, et sert comme déclaration de fin du fichier.
8. **Bloc additif :** Pour fournir une solution à une éventuelle extension dans la définition d'un fichier GIF, un bloc de données peut être ajouté avant chaque descripteur d'image ou avant la fin du fichier.

2. Compression du format GIF :

La compression des images GIF est basée sur l'algorithme de compression LZW. Etant donné que ce format ne permet d'utiliser que 256 couleurs maximum, l'application d'une telle compression sur ces données permet d'avoir des fichiers de tailles très petites par rapport à d'autres formats [GUI 96].

3. Caractéristiques et applications :

Le format GIF est caractérisé par :

- Un fort taux de compression : il permet d'avoir des images de 10 à 30 fois plus petites par rapport au format BMP.
- Il permet d'avoir des images animées en stockant plusieurs images dans le même fichier, en plus dans la version 89a il permet en outre l'entrelacement des images, et l'utilisation des couleurs transparentes, ce qui permet d'obtenir des effets particulièrement intéressants lors de l'affichage de pages WEB [CLE 99][KIV 96].

- En effet si le format GIF permet d’avoir des images de petites tailles c’est grâce à son inconvénient majeur qui est l’utilisation de 256 couleurs maximum. Donc ce format est souhaitable uniquement pour les images de 256 couleurs maximum (ex : les logos). [CLE 99].
- Ce format a un autre inconvénient c’est qu’il n’est pas libre de droit, parce qu’il utilise l’algorithme de compression LZW, tous les éditeurs de logiciel manipulant des images GIF doivent payer une redevance à la société détentrice des droits, Unisys. C’est une des raisons pour lesquelles le format PNG est de plus en plus choisi, au détriment du format GIF [ENC 05].

d. **Format PNG :**

Le format PNG (*Portable Network Graphics*) appelé souvent le format ‘ping’, est mis au point en 1995 par un groupe de volontaires enthousiastes, et qui a eu comme objectif principal de remplacer le format GIF dont son utilisation se fait avec paiement des droits d'utilisation à la société Unisys [SOH 03] [ENC 05].

Il existe en fait deux types d'image PNG :

- PNG-8 : utilise une palette de 256 couleurs, et une compression sans perte. Cette version est le concurrent direct du format GIF.
- PNG-24 : Dans cette version la couleur de chaque pixel est codée sur 3 octets. Et utilise une compression n'est pas dégradante. Ce type peut être considéré comme concurrent du format JPEG [SOH 03][JOH 99].

1. **Structure d’un fichier PNG :**

Un fichier PNG est constitué d'un entête appelé signature, suivi d'une série de segments (appelés *Chunks*) [JOH 99].

1. **Entête de fichier (Signature) :** Cette signature est codée sur 8 octets, contenant respectivement les valeurs suivantes en hexadécimal : 89, 50, 4E, 47, 0D, 0A, 1A, 0A. Cette signature correspond à « PNG » et permet de signaler qu'il s'agit d'un fichier PNG.

2. **Les segments (Chunk) :** Chaque segment est structuré comme suit :

Taille du segment	Type du segment	Données du segment	CRC
-------------------	-----------------	--------------------	-----

Figure 1.3: Structure d'un segment (chunk)

- La taille du segment: codée sur 4 octets, décrivant la taille du segment.
- Le type de segment (chunk type): codé sur 4 octets contenant des caractères ASCII alphanumériques permettant de qualifier la nature du segment.
- Les données du segment (chunk data): codées sur un nombre illimité d’octets.
- Le CRC (Cyclic Redundancy Check): c’est un code de 4 octets permettant de vérifier l’intégrité du segment.

Un fichier PNG comporte plusieurs types de segments, qui peuvent être présents dans n'importe quel ordre, sauf qu'ils doivent commencer par le segment d'en-tête (IHDR chunk) et finir par le segment de fin (IEND chunk) [JOH 99].

Les segments sont répartis en deux classes selon leur importance:

- Les principaux segments (*Critical Chunks*): se sont les segments qui doivent être apparaître obligatoirement dans le fichier (*IHDR*, *PLTE*, *IDAT*, *IEND*), et qui sont utilisés pour la description des parties principales dans le fichier.
- Les segments auxiliaires (*Auxiliary Chunks*) se sont les segments qui peuvent être omis dans la description et qui servent à décrire des parties secondaires dans le fichier [SOH 03].

2. Compression du format PNG :

Ce format est compressé selon l'algorithme LZ77, qui est un algorithme libre de droit, et qui peut produire des images de 10 % à 30 % plus légères que celles en GIF [JOH 99].

3. Caractéristiques et applications :

Le PNG est un format ouvert, qui accepte l'ajout de n'importe quelle particularité (nom d'auteur, signature,...), ces informations peuvent être affichables ou non affichables, ce qui sert à l'identification de l'image.

Ce format vise à remplacer le format GIF, non seulement il assure les mêmes fonctions que ce dernier, il permet en outre :

- D'enregistrer des images de couleurs codées sur 24 bits.
- D'utiliser plusieurs niveaux de transparence de 1 à 256, de complètement opaque à complètement transparent
- D'insérer dans le fichier des informations de correction gamma tout comme dans le format TIFF.

Ces fonctionnalités sont assurées tout en gardant une taille du fichier raisonnable, ce format est appelé à devenir le futur standard d'Internet [SOH 03] [JOH 99].

e. Format PCX :

Le format PCX est un format bitmap créé par **ZSoft Corporation** qui permette d'encoder des images dont la dimension peut aller jusqu'à 65536 par 65536 points, codées sur 1, 4, 8 ou 24 bits. Les données peuvent être enregistrées telle quelle ou compressées suivant l'algorithme RLE. Contrairement au fichier BMP, les fichiers PCX sont la plupart du temps compressés [ENC 05].

1. Structure d'un fichier PCX :

Un fichier PCX est constitué comme suit:

- 1. Entête du fichier :** codé sur 128 octets, et fournit les informations concernant la version du fichier, la méthode de compression, le nombre de

bits par pixel, les coordonnées et les dimensions de l'image, et la résolution de l'écran. [ENC 05].

2. **Données de l'image (*Raster*)** : Les données de l'image sont codées en commençant par le pixel le plus haut à gauche, et en allant de gauche à droite et de haut en bas.
3. **Information** : c'est un octet de séparation entre les données de l'image et celles de la palette de couleurs, et est toujours égale à 192.
4. **Palette de couleurs** : Cette palette est optionnelle, elle n'existe que si le nombre de couleurs est inférieur ou égal à 256 couleurs [ENC 05].

2. Compression du format PCX :

Contrairement au format BMP, ce format est presque toujours compressé suivant l'algorithme RLE, cependant si l'image ne contient pas assez de couleurs identiques contiguës, cette compression risque d'être onéreuse. Pour cette raison une solution a été adoptée, qui consiste à coder les valeurs qui sont inférieures ou égales à 192 et qui ne sont pas redondantes telles quelles (sans facteur de répétitivité), et les valeurs qui sont supérieures à 192 doivent être codées sur deux octets. Par contre les valeurs redondantes sont précédées par le nombre de fois de répétitions plus 192 [ENC 05].

3. Caractéristiques :

Ce format représente le format de fichier graphique le plus ancien, et le plus reconnu et utilisé sous MS-DOS. Actuellement ce format n'est plus presque utilisé car la firme qui l'a créé s'est tournée vers les formats de fichiers en mode points de windows.

Les données sont quasiment tout le temps compressées. Les octets qui ont une valeur supérieure à 192 sont codés sur deux octets, ce qui risque d'augmenter la taille du fichier après compression [ENC 05].

f. Format TIFF :

Le format d'image TIFF (*Tagged Image File Format*) est créé initialement par *Aldus Corporation* en 1986, qui a été acheté ensuite par Adobe [CLE 99]. La première version qui a commencé à se faire connaître fut la version 4.0 à partir d'avril 1987, ce format connaissait ensuite plusieurs améliorations surtout en ce qui concerne la compression et les plages de couleurs utilisées [MAR 03].

1. Structure d'un fichier TIFF :

Le principe du format TIFF consiste à définir des balises (tags) décrivant les caractéristiques de l'image. Ces balises permettent de stocker diverses informations concernant l'image : les dimensions, le nombre de couleurs utilisées, le type de compression, ou bien la correction gamma.

Un fichier de format TIFF est structuré de la manière suivante :

- a. **L'entête du fichier** : constitué de 8 octets qui contiennent des informations concernant l'ordre¹ des octets, la version du format, et le saut vers la première structure des tags (*IFD : Image File Directory*) [ADO 92].

¹ *Little-endian byte order*: les entiers formés de 16 ou 32 bits sont stockés en allant de l'octet le moins fort vers le plus fort, et *Big-endian byte order*: le contraire.

- b. **Structure des tags (IFD : Image File Directory) :** Cette structure peut être à n'importe quel endroit dans l'image, mais pas avant l'entête de l'image. Une image TIFF peut comporter plusieurs IFD, chacune d'elles est structurée comme suit :
- Les deux octets premiers : indiquent le nombre d'entrées dans la structure.
 - Une séquence d'entrées (chacune formée sur 12 octets) contient des informations concernant la valeur et le type du tag, le nombre de valeurs du même type, et la valeur du tag ou l'offset de cette valeur si celle-ci est codée sur plus de quatre octets.
 - Les quatre derniers octets indiquent le saut (l'offset) vers la prochaine structure si elle existe, sinon ce champ contient la valeur 0.
- c. **Données de l'image:** Les données de l'image peuvent être compressées ou non compressées, et codées sur un ou plusieurs morceaux (*Strips*) comme indiqué plus haut, et n'importe où dans le fichier après l'entête.

2. Compression du format:

La compression utilisée pour ce genre de fichier est souvent la compression LZW, c'est la raison pour laquelle ces images sont souvent volumineuses. La compression JPEG est utilisée dans des cas particuliers.

Le type de compression est déclaré dans la structure des tags par le tag 103 en hexadécimal, ainsi si cette valeur est égale à 1 les données sont non compressées.

3. Caractéristiques et application :

Ce format est orienté vers les professionnels (imprimeurs, publicitaires...) car il a l'avantage d'être reconnu sur tous les systèmes d'exploitation : Windows, Mac, Linux, Unix ...

Il permet d'obtenir des images de très bonne qualité (de couleurs codées sur 2 bits jusqu'à 24 bits). Sa portabilité (indépendance du point de vue de la plate-forme) permet une simplification de l'échange de données [CLE 99].

Son principal inconvénient est qu'il produit des images relativement volumineuses, même si elle est inférieure à celle des fichiers BMP [ABA 01].

III-2- Formats vectoriels:

a. Format SWF :

Le format SWF (*ShockWave and Flash file format*) prononcé 'swiff' est développé au début des années 90 par *Jonathan Gay* un américain passionné des jeux informatiques, mais ce format n'a connu de succès qu'après son achat par l'éditeur *Macromedia* en fin 1996 [SOH 03][LAV 06]. Il permet de produire des images vectorielles animées.

1. Structure d'un fichier SWF :

Un fichier SWF est constitué d'un entête suivi d'une séquence de tags: [MAC 04] [LAV 06].

1. **Entête du fichier** : contient la signature 'FWS', la version, et la taille du fichier, ainsi que la taille de chaque cadre, le nombre de cadre (*frame*) par seconde et le nombre total des cadres.
2. **Structure d'un tag** : Chaque tag commence par un entête appelé 'code' qui donne le type et la taille de celui-ci.

Il existe deux types de tags :

- **Les tags de définition** : utilisés pour définir le contenu du fichier (formes, textes, bitmap et son), ces éléments sont appelés caractères et sont stockés dans une structure appelée 'dictionnaire'.
 - **Les tags de contrôle** : utilisés pour manipuler chaque caractère dans le dictionnaire, et sont classés en 3 catégories : les tags d'affichage, les tags de contrôle et les tags des actions.
3. **Dictionnaire** : Le dictionnaire est une séquence de formes (caractères) défini chacun par son identificateur.

Caractère1 (cercle)
Caractère2 (son)
Caractère3 (texte)

Figure 1.4 : Structure d'un dictionnaire

2. **Compression du format SWF :**

L'utilisation du dictionnaire permet de stocker un élément une seule fois et de le réutiliser plusieurs fois ce qui permet un gain énorme dans la taille du fichier. Aussi concernant le mouvement d'un objet, il n'est pas nécessaire de stocker sa position à chaque fois, mais il suffit seulement de stocker son changement [MAC 04].

3. **Caractéristiques et applications :**

Le format SWF est destiné au stockage de l'image animée, généralement il donne des bons résultats soit pour la qualité ou pour la taille du fichier surtout pour les images qui n'ont pas de petits détails [MAC 04].

Pour les images de petite taille ou contenant des petits détails (caractères), ce format donne des fichiers plus lourds par rapport à d'autres formats et plus particulièrement au format GIF [SOH 03].

b. Format SVG :

Le format d'image **SVG** (*Scalable Vector Graphics*) est le fruit des travaux d'une commission rassemblant les principaux éditeurs de logiciels graphiques (Adobe, HP, IBM, Macromedia, Microsoft, Netscape, Quark, Sun, Visio, etc.) qui ont été invité par le W3C au début de l'année 1998 afin de trouver solution pour l'introduction de l'image vectorielle dans le Web. Ainsi ce format a vu le jour au début de l'année 1999, et a été utilisé premièrement par IBM puis par les éditeurs **Corel et Adobe** [SOH 03].

En effet le SVG est un métalangage du type XML, qui utilise des balises et des termes définis pour la description et l'affichage de l'image.

1. Structure d'un fichier SVG :

Étant donné que SVG est un langage pour la description des graphiques bidimensionnels en XML. Un fichier SVG peut être divisé en trois parties :

1. **Entête du fichier :** La première ligne du document sert à identifier le contenu à suivre comme tout document XML.

Exemple :

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

2. **Déclaration de la DTD :** Pour tout document qui doit respecter une DTD (non 'standalone') il faut indiquer l'emplacement de cette dernière, pour les documents non liés à une DTD cette partie est omise.

Exemple :

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/PR-SVG-20010719/DTD/svg10.dtd">
```

3. **Racine du contenu:** Chaque document SVG contient au moins la balise '<svg>'. Cet élément est indispensable car il englobe tous les éléments graphiques de l'image SVG ainsi que sa description. Il commence toujours par la balise '<svg>' et termine par '</svg>'. Entre ces deux balises le document contient :

- La description de l'image elle-même: sa largeur, sa hauteur ...
- La description de tous les éléments graphiques constituant l'image (carré, cercle,...)

Exemple :

```
.1 <svg xml:space="preserve" width="506px" height="310px" viewBox="0 0
.2 2530 1550" enableZoomAndPanControls="true" style="shape-
.3 rendering:geometricPrecision;text-rendering:geometricPrecision;image-
rendering:optimizeQuality">
<g id="ceGroupe"> 'la description de l'élément graphique'</g> </svg>
```

2. Compression des images SVG:

La compression a été introduite sur ce format par l'éditeur Adobe pour réduire son volume qui est souvent un peu élevé. Étant donné qu'un fichier SVG est constitué d'un ensemble de caractères, la compression appliquée sur celui-ci doit être sans perte, sinon l'image risque d'être non lisible [ENC 05].

3. Caractéristiques et applications :

Étant donné que ce format est constitué d'un ensemble de caractères ASCII (textes), ceci permet une modification facile de l'image en utilisant n'importe quel éditeur de texte. En plus le fait que ce format est basé sur un langage XML, ceci rend l'intégration d'une image SVG dans une page WEB non seulement possible mais se fait d'une manière très facile. Il permet en outre de construire des images animées, et interactives [ENC 05].

L'inconvénient majeur de ce format est la taille de l'image qui reste un peu élevée par rapport à d'autres formats vectoriels.

c. Format FIF:

Le format FIF (*Fractal Image File*) est un format d'image de type vectoriel. Ce format est très peu utilisé parce qu'il est breveté. Il permet d'atteindre des taux de compression très élevés sans trop détruire l'image, et ceci en utilisant la compression fractale.

D'autre part, la taille de l'image n'a aucune influence sur celle du fichier, il est possible de zoomer dans l'image, tout en conservant une excellente définition [LEC 06].

III-3- Formats mixtes:

a. Format WMF :

Le format d'image **WMF** (*Windows Meta File*) est un méta fichier, développé par *Microsoft Corporation* [KIV 96], utilisé pour le stockage des images bitmaps ou vectorielles soit dans la mémoire (*clipboard*) ou bien sur le disque. Il est capable de stocker des images de plus de 16 millions de couleurs ; et est reconnu par plusieurs programmes [ABA 01].

1. Structure d'un fichier WMF :

Un fichier WMF est constitué comme suit :

1. **L'entête du fichier :** Contient la taille, le type (fichier sur disque ou dans la mémoire) et la version du fichier, le nombre d'objets, les coordonnées de l'image, ainsi que d'autres octets pour le contrôle et la vérification de l'entête.
2. **Les données :** Un fichier WMF est constitué d'une suite d'objets (jusqu'à 65535 objets au maximum), chacun d'eux est constitué d'un entête qui contient la taille globale de l'enregistrement, le numéro de la fonction et les différents paramètres à passer à cette fonction. Les fonctions sont celles décrites dans le fichier *WINDOWS.H*.

Ainsi les données de l'image sont les données décrivant les différents objets (cercle, carré, bitmap,...), et qui sont définis dans la librairie *WINDOWS.H*. La plupart des objets sont de type vectoriel, ce qui explique le peu de place pris par les fichiers WMF [ENC 05].

2. Compression du format :

Ces fichiers sont souvent constitués à partir des objets de type vectoriel, c'est la raison pour laquelle ces fichiers n'ont pas besoin d'être compressés. Dans le cas d'application de compression celle-ci doit être sans perte.

3. Caractéristiques et applications :

Ce format peut enregistrer des images d'une profondeur de couleurs de 24 bits. Or étant donné qu'il est constitué souvent à partir des objets vectoriels (trait, cercle, carré,...), qui sont caractérisés par un poids très léger par rapport au bitmap, donc ce format a généralement un poids léger sans avoir besoin de compression, ce qui lui donne la possibilité de bien trouver sa place entre les différents formats utilisés.

Il est souvent utilisé pour le stockage des images de bibliothèques qui accompagnent les logiciels de Microsoft [ENC 05] [LEC 06].

b. Format PDF :

Le format PDF (*Portable Document Format*) est un format de fichier qui n'est pas uniquement d'image, il est basé sur le langage *Postscript* niveau 2, il permet de stocker aussi bien des images vectorielles que Bitmap, il est multi-plateforme, MAC, Windows, Unix et Dos grâce au logiciel *Acrobat Reader* largement répandu [ENC 05].

Ce format est né en 1991 d'une idée de *J. Warnock* (l'un des fondateurs de l'éditeur informatique *Adobe*) qui a pensé à créer un format permettant de faire circuler dans l'entreprise des documents numérisés (comportant du texte et des images) consultables quelle que soit la plate-forme utilisée, en préservant la mise en forme, les polices de caractères, les couleurs et les graphiques du document source.

1. Compression du format :

Le format PDF permet la création de fichiers plus compacts qu'en *Postscript*, utilisant les normes de compressions JPEG, LZW et CCITT, facilitant le transfert de gros fichiers sur les réseaux. En effet un fichier Word ou Excel peut diviser son poids par dix lorsqu'il est converti en PDF, tout en comprimant les images qui y sont contenues [ADO 92][ENC 05].

2. Caractéristiques et applications :

C'est un format vraiment très pratique et facile à utiliser, il permet d'avoir des images portables (standards) qui peuvent être lues et imprimées sur différents types d'ordinateurs tout en conservant la mise en page, les couleurs et graphiques d'origine.

L'utilisation de la compression JPEG permet de tirer profit des avantages de celle-ci qui se résume principalement dans le gain énorme dans la taille du fichier tout en gardant une qualité d'image acceptable [ENC 05].

c. Format EPS :

Le format EPS (*Encapsulated PostScript*) par fois appelé *EPSF* (*Encapsulated PostScript Format*) est un format composite, c'est-à-dire qu'il peut contenir du texte, des dessins vectoriels et des images [SOH 03]. Il a été créé par *Adobe Systems*.

Il est basé sur le langage *Postscript* dans sa description des illustrations, c'est pour cette raison qu'il doit être conforme à la DSC (*Convention de Structure des Documents Adobe*) [ENC 05].

1. Structure d'un fichier EPS :

Un fichier EPS contient souvent une image bitmap (en format TIFF ou PICT) de basse résolution utilisée pour permettre une prévisualisation du contenu pour les logiciels qui rencontrent des difficultés à afficher celui-ci, suivie de la description de l'image vectorielle.

Un fichier EPS est constitué au minimum d'un entête contenant la description du fichier, l'organisme développeur et sa version, suivi de la description du contenu (couleurs, compression, illustrations utilisées). Cette description est faite en utilisant le langage *Postscript* [ADO 92][ENC 05].

2. Compression :

La compression a été introduite dans ce format à partir de la version 2, généralement la compression utilisée est une compression JPEG [ENC 05].

3. Caractéristiques et applications :

L'avantage majeur du format EPS est qu'il peut décrire à la fois des illustrations, du texte, des dessins vectoriels, ou des images bitmap, ceci d'une part et de l'autre part il est reconnu par presque tous les logiciels graphiques [ENC 05].

Il s'intègre généralement parfaitement au monde de la PAO puisqu'il est basé sur le langage Postscript. Cependant si ce format est bien adapté et utile pour les transferts, il ne peut être envoyé directement à l'impression, du fait qu'il contient des informations Postscript. En ce qui concerne la taille, ce format reste lourd même après l'introduction de la compression.

d. Format DCS:

Le format DCS (*Desktop Color Separation*) est un format d'image mixte, basé sur le format EPS. Il est développé par *Quark* pour son logiciel de traitement d'image appelé *QuarkXPress* tout en tenant en compte qu'il soit facilement et efficacement indépendant de cette application.[ENC 05].

En effet ce format peut être vu comme une collection de fichier de type EPS. Actuellement les fichiers DCS sont des fichiers EPS conformes à la *DSC*.

1. Structure du format:

La structure d'un fichier DCS est conforme à celle d'un fichier EPS, la seule différence est quelques changements dans la description de l'entête du fichier qui permettent de décrire les données séparées. Un fichier DCS contient une image d'aperçu tout comme le format EPS.

Dans la version DCS1.0 l'image est stockée sur cinq fichiers, le fichier principal est de type EPS qui contient l'image d'aperçu et les lignes de référence aux autres fichiers, quant aux quatre fichiers qui restent, chacun d'eux utilise une extension qui marque la couleur codifiée dans son contenu [ENC 05].

Exemple:

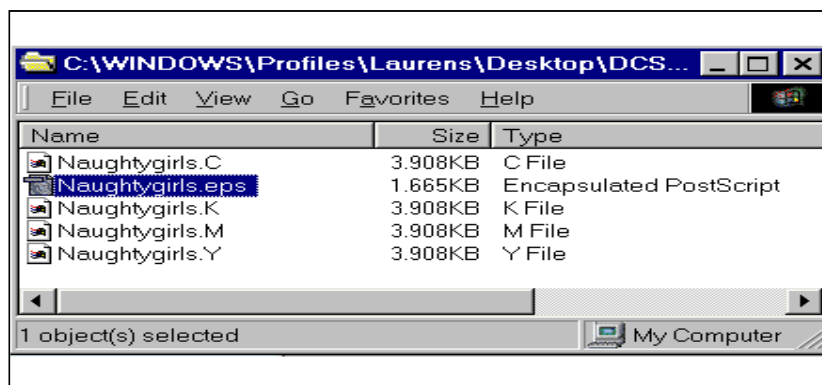


Figure 1.5: Fichiers accompagnant le format DCS

Dans la version DCS2.0 développée entre 1993 et 1995, deux fonctionnalités sont ajoutées à l'ancien format :

- Le choix entre le stockage de l'image sur plusieurs fichiers ou dans un seul.
- La possibilité d'utiliser une table de couleur d'endroit (*spot colour*) en plus des couleurs standards (Cyan, Magenta, Jaune, Noir). Cette fonctionnalité fait de

ce format le format idéal pour les images "*hexachrome*" qui utilisent six couleurs (Cyan, Magenta, Jaune, Noir, Orange et Vert) [ENC 05].

2. Compression:

La compression utilisée dans ce format est la compression JPEG que ce soit pour les images codées sur des fichiers séparés ou dans un seul fichier.

3. Caractéristiques et applications:

L'avantage majeur de ce format est la séparation des couleurs, ce qui permet aux applications de traitement d'image intéressées par cette fonctionnalité d'avoir des bons résultats dans un temps très précis. Cependant avec la rapidité des microprocesseurs d'aujourd'hui cet avantage ne trouve plus sa place.

Vu le grand volume des fichiers résultants de ce format, les deux applications les plus populaires dans l'utilisation de ce format qui sont *QuarkXPress* et *PageMaker* utilisent des images d'aperçu de petite résolution (72 dpi) ce qui pose des problèmes dans l'impression. C'est la raison pour laquelle ce format n'est plus utilisé presque [ABA 01].

e. Format PCT:

Le format PCT ou PICT (*Picture*) développé en 1984 par *Apple Computer (Macintosh)* constitue un standard, il est utilisé comme format d'échange (presse-papier) entre les différentes applications Macintosh. Il est reconnu pour sa capacité de compression des blocs qui n'ont pas beaucoup de variations de couleurs (noir et blanc ou en tons de gris) [ENC 05]. Cependant, il est à proscrire dans le cas des impressions couleurs.

L'ouverture d'une image PCT sur PC se révèle problématique avec de nombreux logiciels, elle peut être ouverte avec *Quick Time (Picture Viewer)* [LEC 06].

1. Compression:

La compression utilisée pour la partie bitmap est de type RLE, dans la version *QuickTime V2.0* la compression JPEG est possible aussi.

2. Caractéristiques et application:

L'utilisation principale de ce format est dans le transfert de fichier entre les applications tournant dans le système MAC.

Son problème réside dans les images qui ont beaucoup de variations de couleurs, qui restent volumineuses même après compression.

Ce format est en cours de disparition du fait que macintosh s'est basculée vers l'utilisation du format PDF [ENC 05].

IV- Comparaison:

IV-1- Comparaison entre les formats étudiés:

Le tableau 1.1 donne un résumé de toutes les caractéristiques et les domaines d'utilisation des formats étudiés dans les sections précédentes où la notation SC signifie sans compression, et 16m signifie 16 millions de couleurs.

Format d'image	Type (B,V)	Qualité	Volume	Couleur	Compression	Volume/Qualité	Applications/caractéristiques
BMP	B	TB	Souvent énorme	RVB (16 m)	SC/RLE	Moyen	Traitement d'image Stockage
JFIF	B	Souvent bonne	Petit	RVB (16 m)	JPEG	(Selon le taux de compression)	Web, transmission (entrelacement possible)
GIF	B	Moyenne	Petit	256	LZW	Moyen	Web (animée + entrelacement)
PNG	B	Bonne	Petit	RVB (16 m)	LZ77	Bien	Web (ouvert: ajout de signature possible)
TIFF	B	TB	Souvent énorme	CMJN (16m)	LZW / JPEG	Moyen	Traitement d'image Archivage (scanner)
PCX	B	TB	Souvent énorme	16 m	RLE/SC	Moyen	N'est plus utilisé
EPS	Mixte	Bonne	Très lourd	CMJN (16m)	Souvent JPEG	Mauvais	Graphisme Transfert PAO
WMF	Mixte	Bonne	Souvent léger	16 m	Souvent Sans compression	Moyen	Format de clipboard Windows
PDF	B	TB	Moyen (selon la compression)	16 m	JPEG / LZW / CCITT	Bien	Multi plateforme transfert / réseau
SWF	V	Bonne	Souvent léger	16 m	SC	Bien	Animée (Web) (macro media flash)
SVG	V	Moyenne	Elevé	16 m	Sans perte	Moyen	Modification facile Bonne intégration en Web
DCS	Mixte	Bonne	Gigantesque (5 fichiers)	CMJN	JPEG	Mauvais	Traitement d'image (presque non utilisé)
FIF	V	Bonne	Petit	16 m	Fractale	T.Bien	Peu utilisé (breveté)
PCT	Mixte	Bonne	Moyen	RVB 16m	RLE/JPEG	Moyen	Format de clipboard pour Macintosh (En cours de disparition)

Table 1.1. Etude comparative entre les différents formats

Ce tableau montre que plusieurs facteurs contribuent à la comparaison des formats de fichiers. En effet il n'existe pas de format idéal dans tous les domaines et avec toutes les applications, mais chacun possède ses domaines d'application souhaitables.

Le facteur le mieux placé à être utilisé dans la comparaison des formats est bien le facteur qualité/poids (qualité d'image par rapport au volume du fichier résultat). Ainsi à partir du tableau on peut remarquer que les formats JFIF, GIF, PNG, PDF et

SWF sont les meilleurs formats qui arrivent à faire un bon compromis entre ces deux facteurs, et c'est la raison pour laquelle ces formats sont préférés sur le Web. Cependant dans le domaine du traitement d'image ces formats sont moins souhaités que d'autres (BMP, TIFF, DCS) du fait que la qualité d'image produite est moins bonne que ces derniers [VIA 00].

Les formats vectoriels sont beaucoup plus souhaités par les logiciels faisant du graphisme, que ceux qui font du traitement d'image. Certains d'eux peuvent être intégrés facilement dans le Web du fait qu'ils sont basés sur des descriptions textuelles.

En général ces caractéristiques sont liées directement au type du format (Bitmap ou vectoriel) et à la technique de compression appliquées. Les avantages et les inconvénients qui caractérisent ces deux types de format sont résumés dans ce qui suit:

IV-2- Formats Bitmaps:

Les images matricielles sont caractérisées par :

a) Les avantages:

- Faciles à créer et à stocker du fait de leur manipulation comme des tableaux.
- Facilité dans la recherche d'informations et la modification des points.
- Bonne adaptation à l'affichage sur écran.
- Possibilité de représenter des images complexes avec une très bonne qualité.
- Une dégradation des données du fichier ne le rend pas inutilisable (sauf pour les données de l'entête) [VIA 00] [HEN 03].

b) Les inconvénients :

- Les images sont souvent volumineuses, ceci rend leur transmission très difficile sur réseaux, et consomme beaucoup d'espace mémoire dans leur stockage. Des méthodes de compression/décompression peuvent alléger ce problème, mais cette solution sera faite au détriment de la qualité et du temps d'affichage de l'image [BRE 03].
- Les opérations de dimensionnement, réduction/agrandissement, changement d'échelle ou rotation sont mal supportées, ce qui entraîne généralement une dégradation (effets d'escaliers) de la qualité d'image.
- L'image à une résolution fixe, elle doit être prévue dès le début pour la résolution du périphérique de sortie (72 dpi pour l'écran et 300 dpi en moyenne pour l'imprimante).

IV-3- Formats vectoriels:

Les formats vectoriels sont caractérisés par:

a) Les avantages:

- Beaucoup adaptés au stockage d'images composées de formes géométriques, avec une possibilité de définition de couches différentes pour les objets [HEN 03][BUR 01].
- Ils supportent facilement les opérations de dimensionnement et adaptation à l'interface de sortie en gardant la même qualité initiale, avec un volume très réduit par rapport au type bitmap [BRE 03].
- La plupart des formats sont en ASCII se qui facilite leur modification.

b) Inconvénients :

- Ils peuvent difficilement stocker des images complexes (réelles), en plus que l'apparence de l'image peut énormément varier en fonction du logiciel qui l'interprète.
- Les fichiers vectoriels sont caractérisés par une intégrité des données, la perte d'une donnée entraîne la perte de l'image toute entière.
- La reconstruction d'une image à partir de vecteurs peut prendre beaucoup plus de temps qu'une image bitmap de complexité égale [BRE 03].

V- Conclusion:

L'information visuelle a pris une grande partie dans notre vie actuelle du fait de son excitation et de son efficacité à mieux exprimer une scène. L'image numérique - étant une de ces deux types d'information (image et vidéo)- a été le sujet de plusieurs recherches depuis plusieurs décennies afin de trouver la bonne méthode à faire représenter celle-ci en la gardant en bonne qualité et en minimisant son volume.

Dans ce chapitre nous avons fait le tour sur quelques formats d'images qui contribuent avec d'autres à la codification de l'image numérique. Chacun de ces formats possède ses points forts et ses points faibles qui lui permettent d'être plus adapté à un domaine qu'à un autre. Beaucoup de critères contribuent à la comparaison des formats d'images, le plus important est sans aucun doute la qualité d'image par rapport au volume du fichier qui en résulte. En fait le volume des images bitmaps dépend de ses dimensions, et le nombre de couleurs utilisées dans celle-ci.

Dans le chapitre suivant nous étudions le langage LWDI qui est un langage développé au laboratoire ***LRIA*** de l'***USTHB***, destiné à la description des images sous forme textuelle, en se basant sur la description de son contour externe, ainsi nous montrerons par la suite qu'il est possible de reproduire l'image à partir de sa description.

DESCRIPTION TEXTUELLE D'IMAGES

I. Introduction :

Notre ère est caractérisée par une grande divulgation des documents techniques (électroniques) qui peuvent contenir à la fois du texte, des images, des animations, du son, et de la vidéo. L'image s'impose comme un support et une source d'informations privilégiée, ceci fait susciter un intérêt croissant à faire codifier l'image d'une manière permettant une facilité d'indexation et de stockage [VIA 00][KEE 90].

Dans cet axe de recherche, une méthode de description d'image a été développée au laboratoire L.R.I.A de l'USTHB. Son principe de base consiste à décomposer l'image en régions représentée par une structure de graphe [LAR 07].

Cette représentation est ensuite traduite moyennant un langage appelé **LWDI** (*Language for Writing Descriptors of Images*) développé au même laboratoire, en une description textuelle suivant le format XML.

Dans ce chapitre nous présentons le principe de base de cette méthode ainsi que le langage LWDI (*Language for Writing Descriptors of Image*).

Nous terminons ce chapitre par une brève comparaison de ce nouveau format avec les formats existants.

II. Segmentation d'image:

L'analyse d'image a pour but l'extraction de l'information caractéristique contenue dans une image. Le résultat d'une telle analyse s'appelle très souvent la "description structurelle". Celle-ci peut prendre la forme d'une image ou de toute structure de données permettant une description des entités contenues dans l'image. Par opposition avec la phase d'interprétation, l'analyse tente dans la mesure du possible, de ne pas prendre en compte le contexte [ZHA 03].

Essentiellement, l'analyse de l'image fait appel à la segmentation où l'on va tenter d'associer à chaque pixel de l'image un label en s'appuyant sur l'information portée (niveaux de gris ou intensité de couleur), et sa distribution spatiale sur le support image [JOL 05] [KER 97].

II.1. Définition:

La segmentation d'une image consiste à effectuer un découpage de celle-ci en parties (régions) de telle façon que chacune d'elles doit posséder des caractéristiques propres à elle seule [MOK 92][ALD 00].

Les régions obtenues doivent vérifier les conditions suivantes:

$$\left\{ \begin{array}{l} \forall i, R_i \neq \emptyset \\ \forall i, j, i \neq j, R_i \cap R_j = \emptyset \\ I = \cup_i R_i \\ \forall i, R_i \text{ connexe} \end{array} \right.$$

La segmentation vise à l'extraction des indices visuels. Après de nombreuses années passées à rechercher la méthode optimale, les chercheurs ont compris que la segmentation idéale n'existait pas. On peut même montrer que le problème de la segmentation est le plus souvent un problème mal posé. Etant donnée une image, il existe toujours plusieurs segmentations possibles [FAL 95][GIO 96].

Or la qualité de l'interprétation d'une image dépend fortement des résultats obtenus après la segmentation. Ainsi une bonne méthode de segmentation sera donc celle qui permettra d'arriver à une bonne interprétation. Elle devra donc avoir simplifié l'image sans pour autant en avoir trop réduit le contenu. Entre autres, elle devra éviter les choix irréversibles [JOL 05][ROS 99].

II.2. Approches de segmentation:

La segmentation d'images ainsi définie est un domaine vaste, souvent plusieurs approches sont envisagées dans ce domaine, les plus utilisées sont:

- **Le seuillage:** consiste à segmenter l'image en plusieurs classes en se basant sur son histogramme. Ces méthodes sont efficace si l'histogramme à vraiment des seuils (pics) nettement séparables [FAL 95].
- **La détection de frontières:** Cette approche consiste à retrouver les points de discontinuité de l'intensité lumineuse présents dans une image, qui correspondent très souvent aux frontières entre parties (régions) [JOL 05].
- **La recherche de régions:** Elle consiste à décrire l'image à l'aide des régions pertinentes regroupant tous les pixels respectant une même propriété.
- Autres approches existent aussi comme: la relaxation, la modélisation, la Transformée de *Hough*, et l'approche de la classification [ALD 00].

Dans notre travail, la méthode de segmentation utilisée s'appelle **JSEG**. Cette méthode est développée à l'université de *Californie* (au Laboratoire de Recherche dans la Vision et Traitement d'Image) [YIN 05].

II.3. Méthode de segmentation JSEG:

JSEG est un algorithme de segmentation développé à l'université de Californie par *Yining Deng* et *B.S. Manjunath* [YIN 05] qui travaillent dans un groupe de recherche dans le Laboratoire de Recherche dans la Vision et Traitement d'Image. Cet algorithme constitue une référence dans le domaine de segmentation.

Il procède tout d'abord à une quantification suivie d'une mesure de similarité à plusieurs échelles. Son approche multi échelles lui permette d'obtenir des bons résultats dans les zones texturées [YIN 05].

La première étape consiste à quantifier les couleurs dans l'image dans plusieurs classes qui peuvent être utilisées pour différencier les régions dans l'image sans prendre en considération la distribution spatiale des couleurs. Ensuite les couleurs des pixels sont remplacées par les labels de classes dont ils font partie [YIN 05].

La deuxième étape consiste à appliquer une segmentation basée sur la méthode dite multiéchelle sur l'image résultante et dite "J-Image".

L'avantage de cette méthode est qu'elle peut être appliquée même sur des scènes de vidéo et fournit un bon résultat.

La figure 2.1 montre le résultat de l'application de cette technique sur une image en couleur:



Figure 2.1 : Segmentation d'image avec la méthode JSEG

III. Méthode de description d'image

III.1. Décomposition de l'image en régions

La première étape consiste donc à décomposer l'image en régions. L'image est généralement constituée des projections d'un ensemble d'objets en 3D superposés sur un arrière plan. Cet arrière plan peut avoir une ou plusieurs couleurs.

La segmentation de l'image en utilisant la méthode JSEG donne une nouvelle image dans laquelle le contour de chaque région est coloré par une couleur différente pour le faire apparaître (voir figure 2.1). Les régions peuvent inclure des parties internes ayant des couleurs ou textures différentes produisant ainsi un ensemble de régions internes.

La seconde étape de cette méthode consiste à décrire chacune des régions de l'image (silhouette).

II.1 III.2. Description de silhouette:

1. Définition:

La silhouette d'un objet est le contour fermé qui englobe l'image, obtenu par les techniques de traitement d'image (seuillage, application d'un filtre, soustraction d'image).

2. Décomposition de silhouette:

A toute silhouette on peut associer un polygone convexe la contenant et passant par un nombre fini de ses points d'extrémité.

Ces points d'extrémité correspondent aux points de courbure de sa frontière. Partant de cet enveloppe convexe il existe un unique rectangle d'aire minimum l'englobant et passant par au moins un de ses sommets, quelque soit l'orientation de la silhouette [HAM 03][JOL 05].

Le fait d'englober la silhouette par un rectangle minimum permet de définir un repère de travail invariant par rotation de la silhouette.

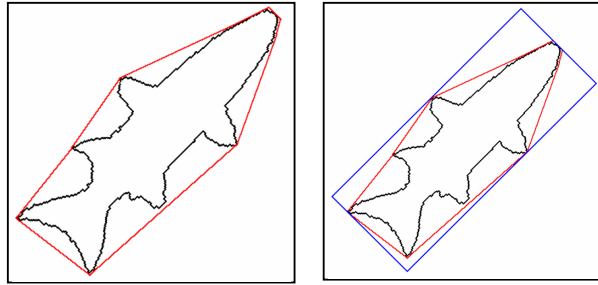


Figure 2.2: Rectangle d'aire minimum

Toute silhouette peut être décomposée en plusieurs éléments (figure 2.3), et ce en suivant les étapes décrites ci-après:

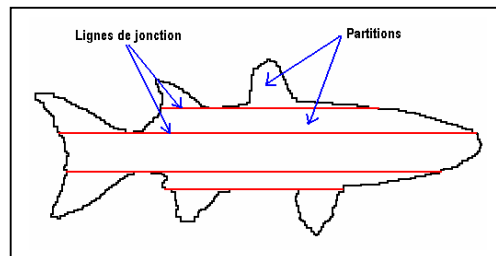


Figure 2.3 : Décomposition de silhouette

- ❖ Tracer le rectangle d'aire minimum englobant la silhouette.
- ❖ Positionner un repère (Oxy) sur le coin le plus bas à gauche de son rectangle d'aire minimum.
- ❖ Balayer le rectangle d'aire minimum en commençant par son point origine O, et ceci suivant l'axe (Ox) ou l'axe (Oy).
- ❖ A chaque passage par un point d'inflexion *haut-bas-haut* dans le cas du balayage suivant l'axe (Oy) ou bien un point d'inflexion *gauche-droite-gauche* dans le deuxième cas, une ligne droite parallèle avec l'axe du balayage (X-ligne ou Y-ligne) est tracée découpant la silhouette en plusieurs parties [LAR 03][HAM 03].

Après le balayage de tout le rectangle d'aire minimum et le découpage de la silhouette on obtient les éléments suivants:

- Chaque partie du contour externe délimitée par une ou deux X-lignes (respectivement une ou deux Y-lignes) est appelée partition.
- Les X-lignes (respectivement Y-lignes) sont appelées les lignes de jonction ou de disjonction selon le nombre de partitions en haut et en bas (respectivement à gauche et à droite). Une ligne est dite ligne de jonction si elle joint plusieurs partitions en haut avec une seule partition en bas, et ligne de disjonction dans le cas contraire.
- Le regroupement de plusieurs partitions (respectivement d'une partition) en haut, d'une ligne de jonction (respectivement d'une ligne de

disjonction) et d'une partition (respectivement de plusieurs partitions) en bas est appelé partie composée [LAR 03].

Ainsi la description de la silhouette consiste décrire chaque élément ainsi que sa position par rapport aux autres.

3. Description d'une partition:

Une partition est constituée de deux frontières: une frontière gauche et une frontière droite. Chaque frontière est décomposée en un ensemble de contours élémentaires délimités par des points de courbures.

Description d'un contour élémentaire:

Un contour élémentaire est la partie du contour externe délimitée par deux points de courbure. Ces points de courbure sont obtenus en appliquant un algorithme de localisation de ces points (ex: algorithme de *Chetverikov*) [CHE 03] tout au long du contour.

Ainsi selon les différents cas possibles du contour externe, un contour élémentaire peut être soit une ligne droite, une courbe convexe, ou bien une courbe concave.

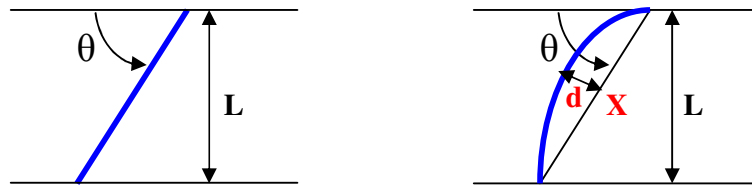


Figure 2.4 : Contour élémentaire

Chaque contour élémentaire est défini par:

- Type du contour: indiquant s'il s'agit d'une ligne droite, courbe convexe, ou une courbe concave.
- Angle d'inclinaison: définit l'angle formé entre la ligne droite passant par les deux points d'extrémité de celui-ci et la ligne horizontale passant par son point haut.
- Degré de convexité/concavité: détermine le rapport d/x (voir figure 2.4) et est calculé uniquement dans le cas d'un contour courbé.
- Longueur: détermine la distance verticale entre les deux lignes horizontales passant par les deux points d'extrémité [LAR 03].

Description de frontière:

La description d'une frontière consiste à décrire l'ensemble des contours élémentaires appartenant à celle-ci dans l'ordre, en commençant à partir du point le plus haut de la frontière.

Les deux frontières gauche et droite d'une partition peuvent avoir le même point de départ et/ou le même point de terminaison ou non, ceci selon sa position si elle est indépendante ou liée à une partie composée [LAR 03].

4. Description d'une ligne de jonction/disjonction:

Une ligne de jonction/disjonction est composée d'un ensemble de segments, ainsi la description de celle-ci consiste à décrire l'ensemble de ces éléments.

Un segment est une partie de la ligne de jonction/disjonction lié soit à une partition en haut ou en bas ou bien à deux partitions une en haut et l'autre en bas.

Un segment est décrit par :

- Type: détermine si le segment est lié avec une seule partition en bas ou en haut ou bien il est lié à deux partitions, une en haut et l'autre en bas.
- Les numéros des partitions liées avec celui-ci.
- Longueur: détermine sa longueur.

5. Description d'une partie composée:

Une partie composée est définie comme étant le regroupement de :

- Plusieurs partitions et/ou parties composées, une ligne de jonction, et une partition en bas

Ou bien:

- Une partition ou une partie composée en haut, une ligne de disjonction, et plusieurs partitions et/ou parties composées en bas.

La figure 2.5 illustre la structure d'une partie composée:

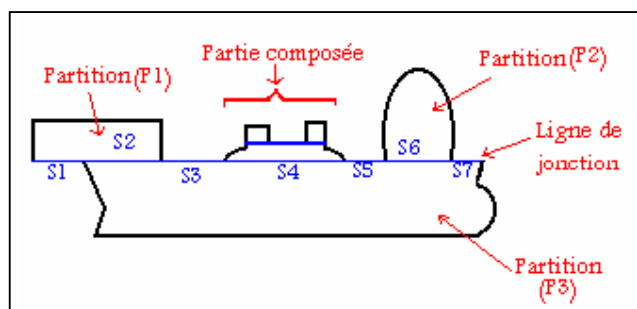


Figure 2.5 : Notion de partie composée

Cette partie composée est constituée de:

- Deux partitions et une partie composée en haut.
- Une ligne de jonction.
- Une partition en bas.

Ainsi la description d'une partie composée consiste à décrire les différentes partitions et/ou parties composées constituant celle-ci ainsi que les lignes de jonction/disjonction.

Cette description doit être faite en commençant par les partitions et/ou les parties composées en haut, la ligne de jonction/disjonction et en fin les partitions et/ou parties composées en bas [LAR 03].

III.3. Le Langage XLWDOS

Le langage XLWDOS (XML *Language for Writing Descriptors Of Silhouette*) est un langage développé au laboratoire LRIA à l'USTHB et proposé pour l'écriture XML de la description de silhouette [LAR 07].

1. Eléments de base du langage:

a. Vocabulaire non terminal:

$V_N = \{ ;Silhouette_i, ;Composed-Part_i, ;Part_i, ;List-Parts_i, ;Left-Boundary_i, ;Right-Boundary_i, ;Contour-Descriptor_i, ;Type_i, ;Length_i, ;Junction-Line_i, ;Inclination-Angle_i, ;Disjunction-Line_i, ;Convexity-Degree_i, ;Concavity-Degree_i, ;Segment-Description_i, ;High-Part-Number_i, ;Low-Part-Number_i, ;NumPart_i, ;NumJunction_i, ;NumDisjunction_i, ;Segments_i \}$.

b. Vocabulaire terminal:

$V_T = \{ s, w, h, <L>, </L>, <R>, </R>, <CP>, </CP>, <P>, </P>, <J>, </J>, <D>, </D> \}$

Les règles de production de ce langage sont :

```

;Silhouettei → <DXLWDOS>
               <Name> "Silhouette name" </Name>
               <CP> ;Parti </CP>
               </DXLWDOS>
;Parti → ;Composed-Parti/
               <P ;NumParti><L>;Left-Boundaryi</L><R>;Right-Boundaryi</R> </P ;NumParti>
;Composed-Parti → ;Parti ;List-Partsi ;Junction-Linei ;Parti/
               ;Parti ;Disjunction-Linei ;Parti ;List-Partsi
;List-Partsi → ;Parti ;List-Partsi / ;Parti
;Left-Boundaryi → ;Contour-Descriptori ;Left-Boundaryi / ;Contour-Descriptori
;Right-Boundaryi → ;Contour-Descriptori ;Right-Boundaryi / ;Contour-Descriptori
;Contour-Descriptori → ;Typei ;Inclination-Anglei ;Lengthi
;Typei → cv ;Convexity-Degreei/cc ;Concavity-Degreei /r
;Inclination-Anglei → INTEGER VALUE
;Lengthi → INTEGER VALUE
;Junction-Linei → <J ;NumJunctioni> ;Segmentsi </J ;NumJunctioni>
;Disjunction-Linei → <D ;NumDisjunctioni> ;Segmentsi </D ;NumDisjunctioni>
;Segmentsi → ;Segment-Descriptioni ;Segmentsi / ;Segment-Descriptioni
;Segment-Descriptioni → s ;High-Part-Numberi ;Low-Part-Numberi ;Lengthi/
               w ;Low-Part-Numberi ;Lengthi / h ;High-Part-Numberi ;Lengthi
;High-Part-Numberi → INTEGER VALUE
;Low-Part-Numberi → INTEGER VALUE
;Convexity-Degreei → INTEGER VALUE
;Concavity-Degreei → INTEGER VALUE
;NumJunctioni → INTEGER VALUE
;NumDisjunctioni → INTEGER VALUE
;NumParti → INTEGER VALUE

```

2. Règles d'écriture d'un descripteur LWDOS:

Pour l'écriture d'un descripteur LWDOS les règles suivantes doivent être suivies:

- Un descripteur LWDOS (description d'une silhouette) est toujours écrit entre les deux balises **<DXLWDOS>** et **</DXLWDOS>**.
- La description d'une partie composée est délimitée par les deux balises **<CP>** et **</CP>**.
- La description d'une partition est délimitée par les balises **<P_n>** et **</P_n>** avec *n* est le numéro de la partition.
- La description de la frontière gauche est délimitée par les balises **<L>** et **</L>**, et celle de la frontière droite est délimitée par les balises **<R>** et **</R>**.
- La description d'une ligne de jonction est délimitée par **<J_n>** et **</J_n>**, et celle d'une ligne de disjonction par **<D_n>** et **</D_n>** avec *n* est le numéro de cette ligne dans la silhouette.

3. Ecriture XML d'un descripteur LWDOS:

Comme tout document XML, un document XLWDOS doit respecter la DTD (*Document Type Definition*) suivante:

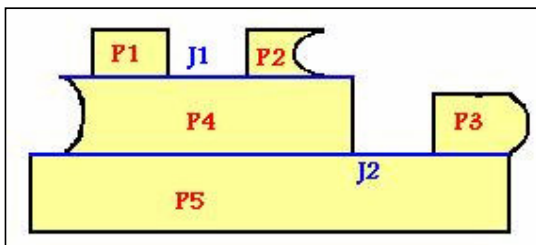
XLWDOS.dtd
<pre><!ELEMENT XLWDOS (CP P)> <!ATTLIST XLWDOS Name CDATA #IMPLIED > <!ELEMENT CP (((CP , P+) (P+ , CP?) (P , P+)) , J , P) ((CP P) , D , P , P+))> <!ELEMENT P (L,R)> <!ATTLIST P num CDATA #REQUIRED> <!ELEMENT L (CV CC LN)+> <!ELEMENT R (CV CC LN)+> <!ELEMENT LN EMPTY> <!ATTLIST LN length CDATA #REQUIRED inclin CDATA #REQUIRED> <!ELEMENT CV EMPTY> <!ATTLIST CV length CDATA #REQUIRED inclin CDATA #REQUIRED></pre>

```

    degree CDATA #REQUIRED>
<!ELEMENT CC EMPTY>
<!ATTLIST CC
    length CDATA #REQUIRED
    inclin CDATA #REQUIRED
    degree 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>

```

4. Exemple de description de silhouette en LWDOS:



```

<DXLWDOS>
<Name> Exemple 3 </Name>
<CP>
    <CP>
        <P1><L> r 90 30 </L> <R> r 180 50 r 90 30 </R></P1>
        <P2><L> r 90 30 </L> <R> r 180 50 cc 35 90 30 </R></P2>
    </CP>
    <J1> w P3 20 s P1 P3 50 w P3 50 s P2 P3 50 w P3 20 </J1>
    <P3> <L> cc 25 90 50 </L> <R> r 90 50 </R></P3>
    </CP>
    <P4><L> r 90 40 </L> <R> r 180 50 cv 25 90 40 </R></P4>
    <J2> w P5 20 s P3 P5 190 w P5 50 s P4 P5 50 </J2>
    <P5><L> r 90 50 r 180 310 </L> <R> r 90 50 </R></P5>
    </CP>
</DXLWDOS>

```

Figure 2.6 : Exemple de description de silhouette

III.2. Description de région:

1. Définition:

Une région est un contour fermé localisé à l'intérieur d'une image [FAL 95]. Cette région correspond au contour externe d'un objet ou à une partie de l'image dans laquelle les pixels ont des couleurs très proches.

Chaque région peut contenir théoriquement un nombre quelconque de régions internes, dont chacune d'elle peut contenir à son tour des régions internes et ainsi de suite, ceci mène à utiliser la notion de niveau de région.

2. Niveau d'une région interne:

La description d'une région interne doit être écrite relativement à sa région parente. Une région de niveau 0 est une région qui ne contient aucune région interne. Récursivement, on attribut à chaque région qui contient une région de niveau inférieur ou égal à k le niveau $k+1$ [LAR 07]. La notation S_{ij} définit la région numéro j du niveau i . La région principale de niveau n est la région qui englobe toutes les régions, ces dernières ont toutes un niveau inférieur à n .

Pour calculer le niveau d'une région on utilisera l'algorithme suivant [LAR 07]:

Algorithme

Begin

- Localiser le cadre contenant la forme.
 - Localiser les silhouettes de niveau 0 et calculer leurs descripteurs LWDOS et leurs positions par rapport au rectangle.
 - Niveau_Actuel:=0
- Tant qu'il y a de forme

Faire

Tant qu'il y a de formes renfermés dans la forme qui à le niveau $\leq$ Niveau_Actuel

Faire

- Localiser la silhouette correspondante
- Calculer son descripteur LWDOS
- Niveau:= Niveau_Actuel + 1

Fin

Niveau_Actuel:= Niveau_Actuel + 1

Fin

Ecrire le descripteur LWDI de la région

Fin

3. Description des régions et régions internes en LWDI:

Le langage LWDI (*Language for Writing Descriptors of Images*) est un langage dérivé du langage XML développé au niveau du laboratoire LRIA à l'USTHB. Il permet la description des images en se basant sur la description des différentes régions (contour externe, couleur, position).

a. Eléments de base du langage:

Vocabulaire non terminal:

Le vocabulaire non terminal est une extension du vocabulaire non terminal du LWDOS.

$$V_N = V_{NXlwdos} \cup \{Shape, Name, Internal\ Shape, List-Internal-Shapes, Name, Color, Position, Silhouette, Red, Green, Blue, X-Position, Y-Position\}$$

Les règles de production:

$iShape_i \rightarrow \langle Shape \rangle \langle Name \rangle iName_i \langle /Name \rangle \langle Color \rangle iPosition_i iSilhouette_i$
 $iList-Internal-Shapes_i \rightarrow iList-Internal-Shapes_i \langle /Shape \rangle$
 $iList-Internal-Shapes_i \rightarrow iInternal-Shape_i iList-Internal-Shapes_i / iInternal-Shape_i$
 $iInternal-Shape_i \rightarrow \langle Internal\ Shape \rangle \langle Name \rangle \text{“name of internal shape”} \langle /Name \rangle$
 $\langle Color \rangle iPosition_i iSilhouette_i$
 $\langle /Internal\ Shape \rangle /$
 $\langle Internal\ Shape \rangle \langle Name \rangle \text{“name of internal shape”} \langle /Name \rangle$
 $\langle Color \rangle iPosition_i iSilhouette_i iList-Internal-Shapes_i$
 $\langle /Internal\ Shape \rangle$
 $iPosition_i \rightarrow iX-Position_i iY-Position_i$
 $iColor_i \rightarrow iRed_i iGreen_i iBlue_i$
 $iRed_i \rightarrow IntegerValue.$
 $iGreen_i \rightarrow IntegerValue.$
 $iBlue_i \rightarrow IntegerValue.$
 $iX-Position_i \rightarrow INTEGER\ VALUE$
 $iY-Position_i \rightarrow INTEGER\ VALUE$
 $iName_i \rightarrow string$

b. Règles d'écriture d'un descripteur LWDI:

La description d'une région doit respecter les règles suivantes:

- La description de la région est délimité par les deux balises $\langle Shape \rangle$ et $\langle /Shape \rangle$, et celle d'une région interne est délimitée par $\langle Internal\ Shape \rangle$ et $\langle /Internal\ Shape \rangle$.
- Le nom de la région est délimité par les deux balises $\langle Name \rangle$ et $\langle /Name \rangle$.
- La couleur est délimitée par les deux balises $\langle Color \rangle$ et $\langle /Color \rangle$.
- La position est déclarée entre les deux balises $\langle Position \rangle$ et $\langle /Position \rangle$. Cette position correspond à la position du rectangle d'aire minimum (coin supérieur à gauche) englobant cette région.[LAR 07][LAR 03]

4. Ecriture XML d'un descripteur LWDI d'une région :

Un descripteur XLWDI d'une région doit respecter la DTD suivante :

ShapeXLWDI.dtd

$\langle !ELEMENT\ Shape\ (Position, Color, XLWDOS, InternalShape^*) \rangle$

$\langle !ATTLIST\ Shape$

```

Name CDATA #IMPLIED>
<!ELEMENT Position EMPTY>
<!ATTLIST Position
    X CDATA #REQUIRED
    Y CDATA #REQUIRED>
<!ELEMENT Color EMPTY>
<!ATTLIST Color
    Red CDATA #REQUIRED
    Green CDATA #REQUIRED
    Blue CDATA #REQUIRED>
<!ELEMENT InternalShape (Position , Color , XLWDOS , InternalShape*)>
Name CDATA #IMPLIED>

```

5. Exemple:

L'exemple suivant montre la description d'une région contenant un ensemble de régions internes.

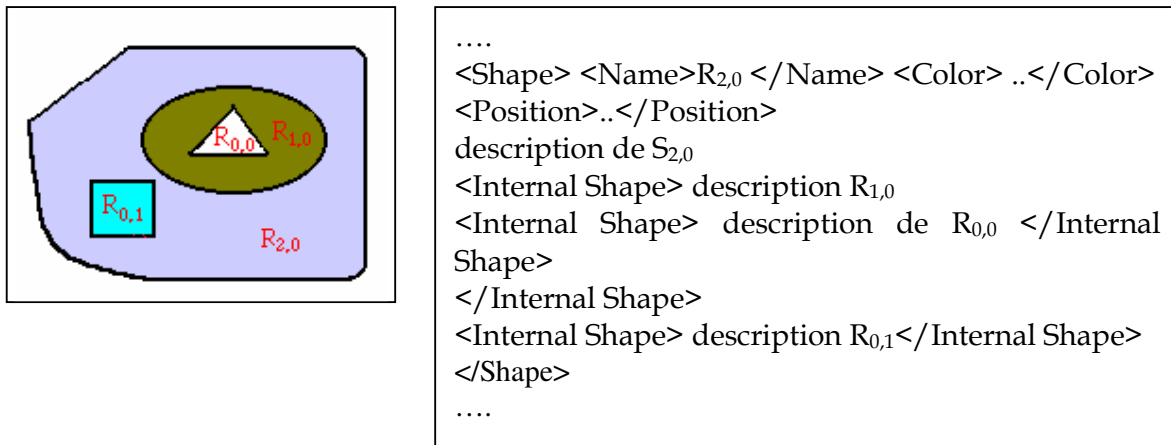


Figure 2.7 : Exemple de description de région

III.3. Description de l'image globale:

L'image donc est composée d'un ensemble de régions dont chacune d'elles peut contenir à son tour des régions internes. Ainsi la description de l'image revient donc à la description du contour de chaque région, ainsi que sa couleur et sa position dans l'image.

En dehors de la description des régions, la description de l'image doit fournir:

- Le nom de l'image.
- Les dimensions de l'image données en pixel.
- La couleur de l'arrière plan en RVB.

a. **Éléments de base du langage:**

Les règles de production sont celles déjà données dans les deux parties précédentes en plus nous utiliserons les règles suivantes :

$S_0 \rightarrow \langle \text{Name} \rangle \langle \text{Color} \rangle \langle \text{Length} \rangle \langle \text{Shape}_1 \rangle / \langle \text{Name} \rangle \langle \text{Color} \rangle \langle \text{Length} \rangle \langle \text{Shape}_1 \rangle .. \langle \text{Shape}_n \rangle$
 $\langle \text{Length} \rangle \rightarrow \langle \text{Width} \rangle \langle \text{Height} \rangle$
 $\langle \text{Width} \rangle \rightarrow \langle \text{IntegerValue} \rangle.$
 $\langle \text{Height} \rangle \rightarrow \langle \text{IntegerValue} \rangle.$

b. **Syntaxe du langage LWDI:**

La syntaxe utilisée par le langage LWDI est indiquée ci-dessous:

- Les balises `<DXLWDI>` et `</DXLWDI>` sont utilisées respectivement au début et à la fin du descripteur.
- Les balises `<Name>` et `</Name>` sont utilisées respectivement pour délimiter la déclaration d'un nom (nom de l'image, d'une région ou d'une silhouette).
- Les balises `<Length>` et `</Length>` sont utilisées respectivement pour délimiter la déclaration des dimensions de l'image. La première valeur indique la largeur de l'image et la deuxième indique sa hauteur, ces valeurs sont exprimées en pixel.
- Les balises `<Background>` et `</Background>` sont utilisées respectivement pour délimiter la déclaration de la couleur de l'arrière plan de l'image. Ces couleurs sont exprimées en RVB.

❖ **Remarques:**

- Une image peut contenir un nombre indéterminé de régions et de régions internes.
- En cas d'absence de couleur les valeurs RGB sont remplacées par le mot 'None'.
- La position d'une région ou d'une région interne donnée dans le descripteur LWDI est calculée toujours par rapport à l'image globale.

c. **Ecriture XML du descripteur LWDI global :**

Le descripteur XLWDI de l'image globale doit respecter la DTD suivante :

XLWDI.dtd
<pre>< !ELEMENT XLWDI (Length , Background , Shape*)> <!ATTLIST XLWDI Name CDATA #IMPLIED> <!ELEMENT Length EMPTY></pre>

```

<!ATTLIST Length
    Width CDATA #REQUIRED
    Height CDATA #REQUIRED>
<!ELEMENT Background EMPTY>
<!ATTLIST Background
    Red CDATA #REQUIRED
    Green CDATA #REQUIRED
    Blue CDATA #REQUIRED>

```

d. Exemple:

```

<DXLWDI>
  <Name> Image Exemple </Name>
  <Length> 300 200 </Length>
  <Background> 155 220 50 </Background>
  <Shape> Description de la Region1 </Shape>
  <Shape> Description de la Region2 </Shape>
  ...
  <Shape> Description de la Régionn </Shape>
</DXLWDI>

```

IV. Caractéristiques du format LWDI:

IV.1. Caractéristiques:

La description d'une image dans le langage LWDI permet d'obtenir un fichier texte structuré, ceci permet de bénéficier généralement des avantages des formats vectoriels, et plus spécifiquement par :

- Etant donné que l'image est sauvegardée sous forme textuelle, ceci permet de faciliter la modification de celle-ci par un simple éditeur de texte.
- Une description invariante dans le cas du déplacement ou de rotation de l'objet, il suffira d'inclure la rotation du rectangle d'aire minimum englobant chacune des régions de l'image.
- Possibilité de tirer des indices invariants de l'image ce qui permet de faciliter la tâche d'indexation de l'image.
- Possibilité d'intégrer cette description dans des pages Web, du fait que le langage LWDI est dérivé du XML.

IV.2.Compression du format:

Un fichier LWDI est un fichier textuel, ainsi la compression qu'on doit appliquer doit être sans perte, sinon le fichier risque d'être erroné.

En effet généralement le fichier de description peut contenir beaucoup d'espaces et de retour à la ligne pour donner une bonne structure du fichier et faciliter sa compréhension par l'homme. Ainsi la compression peut fournir des bons résultats dans ce cas.

IV.3.Comparaison avec d'autres formats:

Pour mieux discerner les avantages de ce format par rapport aux autres formats d'image, le tableau suivant donne le résultat d'une comparaison entre celui-ci et les formats les plus utilisés.

Cette comparaison est faite sur l'image ci-dessous, dans laquelle nous avons utilisé les logiciels "*Paint*" de Microsoft et "*PhotoShop 6.0*" de la maison Adobe pour faire sauvegarder l'image en différents formats:

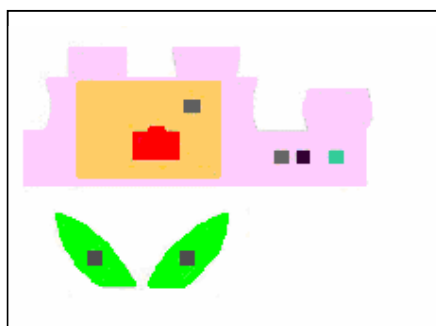


Figure 2.8 : Image de comparaison

Les résultats de comparaison de l'image dans la figure 2.8 sont indiqués dans le tableau suivant:

Format	Taille du fichier (octet)
LWDI (<i>sans compression</i>)	2774
LWDI (<i>compressé avec Winrar</i>)	597
BMP	412182
JPEG	8498
GIF	2716
TIFF	7971
PNG	1920
TGA	51388

Table 2.1. Résultat de comparaison entre l'espace mémoire nécessaire aux différents formats.

Ce tableau montre les volumes des fichiers résultants de la codification de la même image sous formats différents.

Ainsi on peut remarquer qu'en outre que la possibilité d'indexation et d'intégration dans le Web, le format LWDI offre des images avec des tailles très réduites. En effet les tailles des fichiers LWDI non compressées sont généralement aux environs de celles des formats GIF et PNG (256 couleurs). En y ajoutant la compression celle-ci peut diminuer à un cinquième de sa taille originale, ceci lui permet d'être un leader dans ce domaine.

V. Conclusion:

Dans ce chapitre nous avons étudié le langage LWDI qui est un langage inspiré du langage XML, et qui permet de décrire l'image sous forme textuelle. Cette description se base sur la description du contour fermé de chaque région ou région interne nécessitant ainsi une bonne segmentation de l'image.

Cette description à l'avantage d'être d'une part textuelle ce qui facilite sa modification par un simple éditeur de texte, et de l'autre part elle est structurée se qui permet la recherche d'information dans le fichier et de pouvoir indexer ces fichiers dans une base de données.

En fait pour pouvoir tirer profit de cette technique de description, il faut bien disposer des outils (logiciels) permettant de générer cette description à partir de n'importe quelle image, et aussi des outils qui permettent de faire l'opération inverse, c'est-à-dire produire l'image à partir de sa description LWDI.

En ce qui concerne la première opération (générer la description d'une image), celle-ci a été déjà sujets de deux recherches à l'institut d'informatique (USTHB) et qui ont donné deux outils: un pour la description de silhouette (en LWDOS) et l'autre pour la description d'image.

En ce qui concerne la deuxième opération (produire l'image appropriée à une description), celle-ci sera l'objet d'étude dans notre prochain chapitre.

RECONSTRUCTION D'IMAGE A PARTIR DE SON DESCRIPTEUR LWDI

I. Introduction

Les performances obtenues par la description d'images en LWDI sont diverses (possibilité d'indexation, gain en mémoire et en transfert,...), ceci permet à elle d'être utile dans plusieurs domaines, surtout dans les bases de données images, du fait qu'elle donne la possibilité de la recherche d'image en se basant sur un nombre de critères.

Un autre avantage est la possibilité d'être intégrée facilement dans le Web par le codage et le transport d'images moyennant cette description.

Ceci ne peut être utile sans disposer d'outils nécessaires à la reconstruction d'image à partir de cette description. Donc le problème qui se pose est *Comment peut-on produire l'image à partir de sa description LWDI ?*

Dans ce chapitre nous proposons une méthode pour la reconstruction d'images décrites par LWDI ainsi que les algorithmes nécessaires à cette opération.

Etant donné qu'une image est constituée d'un ensemble de régions, dont chacune peut contenir un ensemble de régions internes, ainsi la reconstruction de l'image est basée d'une part sur la reproduction de la silhouette de chaque région, et de l'autre part sur le positionnement de chaque région.

Ceci mène donc à aborder trois éléments:

- ❖ Comment reconstruire une silhouette à partir de sa description textuelle (LWDOS)
- ❖ Comment positionner les différentes régions et régions internes dans l'image?
- ❖ Comment traiter la couleur de l'image globale ainsi que celle de chacune des régions et régions internes?

Nous montrerons d'abord que l'image produite à partir d'une description LWDI est unique. Nous proposons ensuite la méthode et les algorithmes nécessaires à cette reconstruction. Nous terminons par étudier la qualité de l'image reconstruite ainsi que les performances de la méthode proposée.

II. Unicité de l'image reconstruite

Nous montrerons dans cette section qu'il est possible à partir d'une description LWDI de reconstruire l'image appropriée, et que cette image est unique.

II.1. Reconstruction de silhouettes :

Une silhouette peut être constituée soit d'une partition simple (unique) ou d'une partie composée, qui est à son tour constituée d'un ensemble de partitions et de lignes de jonction. Chaque partition est constituée d'un ensemble de contours élémentaires (droit, convexe ou concave). Chaque ligne de jonction est constituée d'une série de segments placés l'un après l'autre.

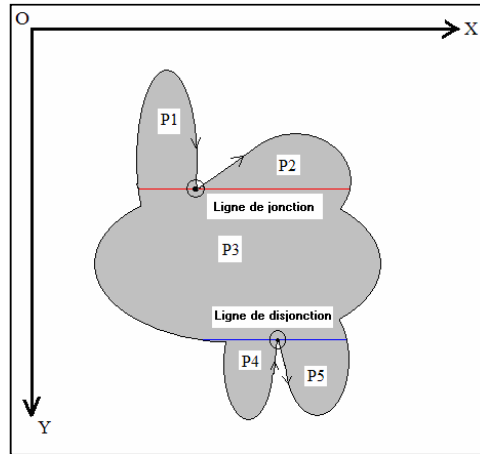


Figure 3.1: Silhouette

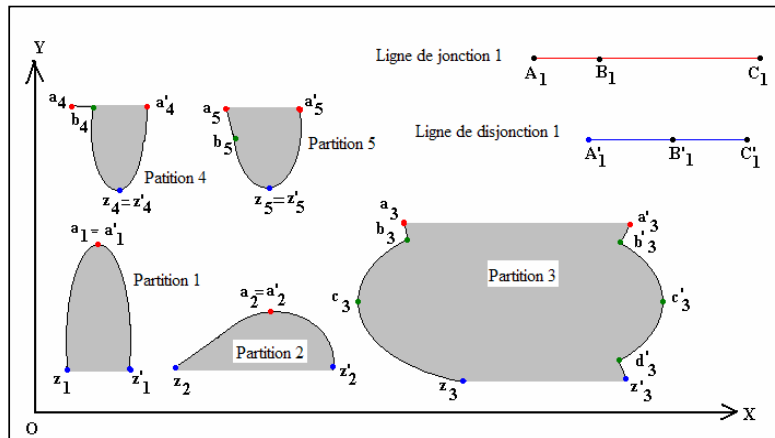


Figure 3.2: Décomposition de la silhouette

La description LWDOS d'une silhouette est basée sur la décomposition montrée dans les deux figures 3.1 et 3.2.

Les informations fournies dans le descripteur LWDOS concernant un contour élémentaire sont : le type ('r' : droit, 'cv' : convexe, 'cc' : concave), l'angle d'inclinaison (' θ '), la longueur ('L') et le degré de convexité ou de concavité ('Dc') dans le cas d'un contour courbé.

Dans le cas d'un contour droit, en connaissant le point de départ Pt_1 du segment, il existe un point ' Pt_2 ' unique qui vérifie les conditions décrites (angle d'inclinaison de la droite ' Pt_1Pt_2 ' = ' θ ' et la longueur = 'L') (voir à gauche dans la figure 3.3).

Dans le cas d'un contour courbé, en connaissant le point de départ ' P_1 ', l'angle d'inclinaison et la longueur, le point d'arrivée peut être facilement calculé et sera unique comme dans le cas d'un contour droit. Pour tracer la courbe qui passe par ces deux points, il existe un nombre illimité de cercles de rayons différents qui passent par ' P_1 ' et ' P_2 ', dont leurs points de centres sont alignés sur la droite qui passe par le point du milieu de la partie droite $[P_1P_2]$ (voir la figure 3.3 l'image à droite).

Pour chaque cercle d'un rayon ' r ' dont son centre est d'un côté de la droite $[P_1P_2]$, il existe un autre cercle dont son centre est de l'autre côté de la même droite et qui a le même rayon ' r '.

Maintenant en connaissant le degré de convexité/concavité ' D_c ' le rayon du cercle ' r ' peut être calculé, et ainsi le nombre de cercles qui vérifient les conditions est réduit à 2 (un cercle en haut de la droite et l'autre en bas).

En connaissant le type de la courbe ' cc ' ou ' cv ' on peut déterminer lequel des deux cercles qui vérifie toutes ces conditions.

Ainsi la courbe qui répond à la description peut être reconstruite d'une manière unique.

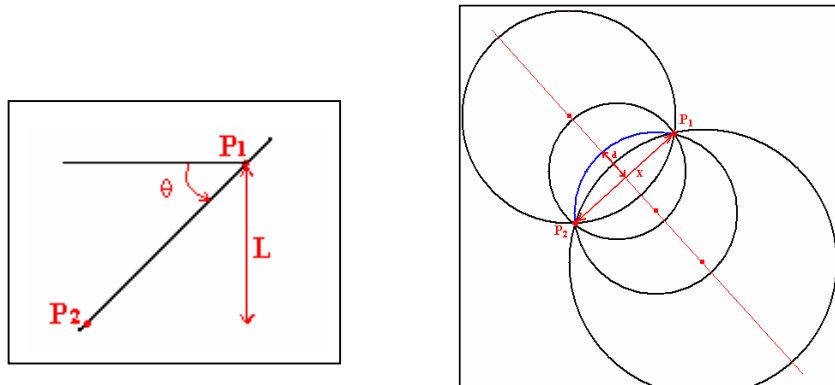


Figure 3.3 : Unicité de reconstruction d'un contour élémentaire

La propagation de cette opération sur l'ensemble des contours élémentaires permet de reconstruire toute la partition. [DAH 06].

En ce qui concerne les segments dans une ligne de jonction, ceux-ci peuvent être considérés comme un cas particulier des contours élémentaires droits (où ' θ '=180), ainsi en connaissant le point de départ et la longueur, il existe un seul point d'arrivée qui vérifie ces conditions. La ligne de jointure est reconstruite en propageant ces opérations sur l'ensemble des segments.

La description d'un segment donne aussi les noms des partitions liées à celui-ci ainsi que leurs positions (en haut ou en bas).

Donc les informations fournies dans la description d'un segment dans une ligne de jonction permettent la reconstruction de celui-ci ainsi que le positionnement des partitions autour de lui.

La coopération de ces deux opérations (reconstruction de partition et reconstruction de ligne de jointure) permet la reconstruction des parties composées.

Ainsi donc la silhouette peut être reproduite intégralement à la description LWDOS.

II.2. Reconstruction d'une région :

Une région est constituée d'un contour fermé, englobant un espace de points de même couleur. Cette région peut contenir des régions internes de couleur différentes.

Cette région peut être décomposée donc en plusieurs régions simples dont chacune est constituée d'un contour externe (silhouette) et d'un fond coloré, et dont la reconstruction de chacune est faite de la même façon.

Le contour fermé C_i englobant la région R_i n'est qu'une silhouette donc sa reconstruction se fait selon la méthode présentée en II.1. Après la reconstruction de la silhouette on procède donc à la colorisation des points à l'intérieur.

Exemple:

Soit la région $R_{2,0}$ dans la figure ci-dessous:

Cette région est constituée de deux régions internes $R_{1,0}$ et $R_{0,1}$. La région $R_{1,0}$ contient elle-même une région interne $R_{0,0}$.

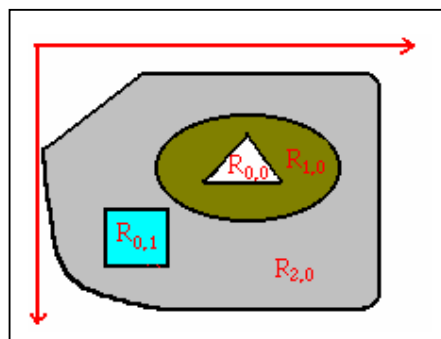


Figure 3.4: Structure d'une région

La décomposition de cette région en plusieurs régions élémentaires est montrée dans la figure 3.5:

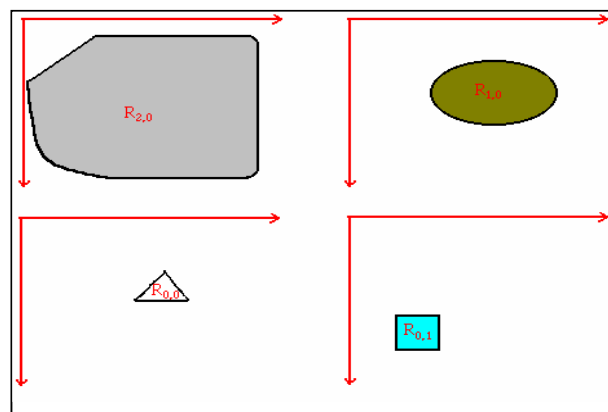


Figure 3.5 : Décomposition d'une région

Ainsi la reconstruction de $R_{2,0}$ est la reconstruction de l'ensemble de régions internes $R_{1,0}$, $R_{0,0}$ et $R_{0,1}$, chacune dans sa position exacte.

II.3. Reconstruction de l'image :

L'image est constituée d'un rectangle englobant un fond de couleur Cl , et d'un ensemble de régions $R = \{R_0, R_1, \dots, R_n\}$.

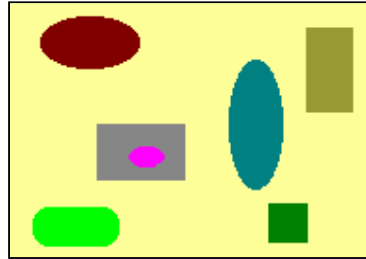


Figure 3.6 : Structure d'une image

L'image peut être donc décomposée en une couleur du fond et une suite de régions superposées sur ce dernier (figure 3.6).

Les régions peuvent être reconstruites en propageant le processus de reconstruction sur l'ensemble des régions une par une.

L'image donc est reconstruite en commençant par l'arrière plan, et ensuite les régions une par une. Chaque région doit être reconstruite dans sa position exacte par rapport à l'image globale.

III. Méthode de reconstruction d'image:

III.1. Schéma général de la méthode

Le schéma général de la méthode proposée pour la production de l'image à partir de son descripteur LWDI est représenté par la figure 3.7.

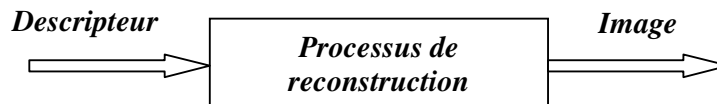


Figure 3.7: Schéma général

Le processus de reconstruction d'image reçoit en entrée un descripteur LWDI, l'analyse et extrait les informations nécessaires et suffisantes pour la reconstruction et visualisation de chaque composant de l'image (régions et régions internes)

Étant donné qu'une image est constituée d'une couleur de fond, et d'un ensemble de régions, dont chacune d'elles est constituée d'une silhouette représentant son contour externe, d'une couleur de fond et d'un ensemble de régions internes (une région interne a la même structure qu'une région).

Ainsi ce processus peut être représenté par le schéma suivant:

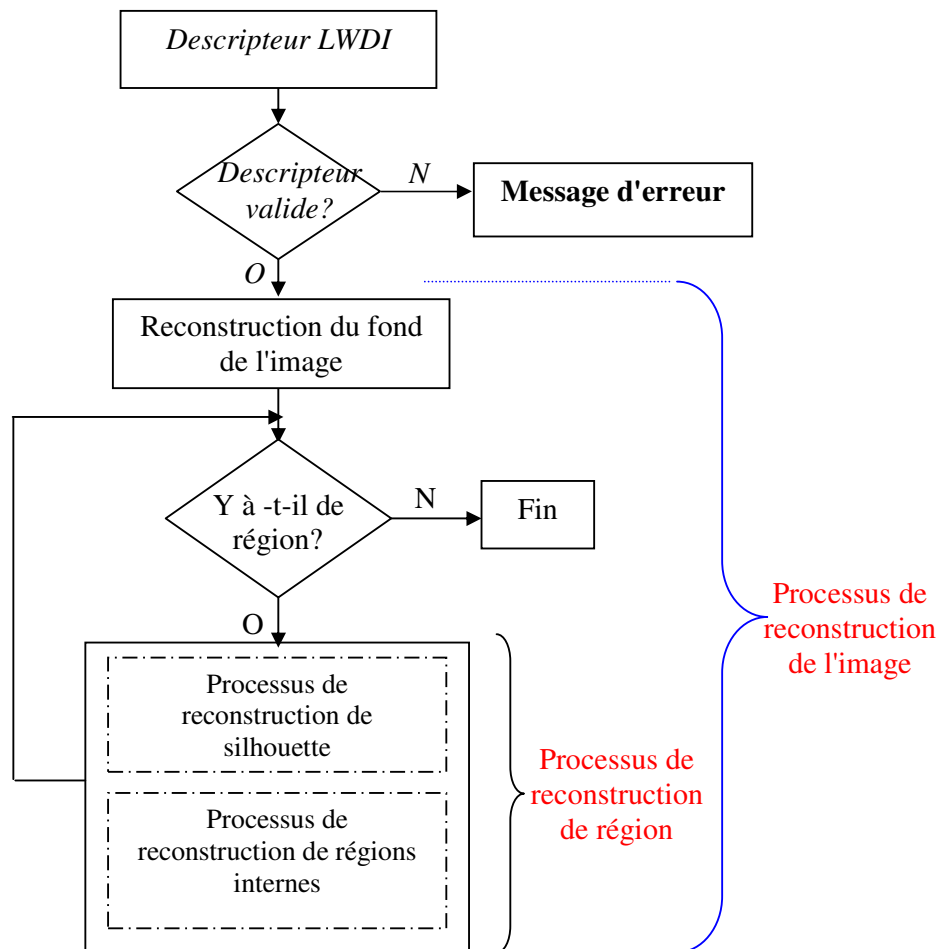


Figure 3.8 : Schéma général du traitement

La reconstruction de l'image se base sur quatre processus principaux:

-Vérification du descripteur: ce processus consiste à prendre en entrée le descripteur LWDI, à vérifier sa syntaxe, et à déclencher le processus de reconstruction de l'image dans le cas où le descripteur est jugé valide.

-Reconstruction de l'image globale: ce processus est le processus père (ancêtre). Il commence d'abord par l'extraction des données nécessaires à la reconstruction du fond de l'image (dimensions et couleur de l'image), le "processus de reconstruction de région" est ensuite lancé pour chacune des régions internes.

-Reconstruction de région: ce processus consiste à reconstruire le contour externe de la région (Appel au processus de reconstruction de silhouette) tout en prenant en compte sa position et sa couleur dans l'image. Ce processus est exécuté d'une manière récursive sur toutes les régions internes.

-Reconstruction de silhouette: c'est le processus qui consiste à calculer les points de démarrage et de terminaison de chaque contour élémentaire constituant la frontière de la silhouette et à dessiner chacun d'eux dans son emplacement exact. Ces emplacements sont calculés par rapport à l'image globale.

III.2. Principes de base et Algorithmes

1) Reconstruction de silhouette:

Une silhouette est constituée soit d'une seule partition ou bien d'une partie composée, qui peut être à son tour constituée de plusieurs parties composées.

Une méthode de reconstruction de silhouette à partir de sa description LWDOS a été déjà proposée par A. Dahmane [DAH 06].

La méthode de la reconstruction de silhouette que nous proposons est similaire à celle proposée par A. Dahmane [DAH 06]. Quelques changements ont été apportés concernant les informations qui doivent être calculées à chaque lecture de la description d'un composant de la silhouette, et concernant l'ordre du traitement des partitions dans les parties composées.

Au lieu de calculer le point le plus loin comme a été proposé dans la méthode citée ci-dessus, nous proposons de calculer le point du centre du cercle qui passe par le contour courbé, en effet ceci facilite considérablement la recherche du point le plus à gauche dans la courbe qui doit être utilisé dans le calcul de position de la silhouette par rapport à son rectangle d'aire minimum.

En ce qui concerne l'ordre de reconstruction des éléments dans une partie composée, nous avons choisi de commencer la reconstruction d'une silhouette toujours à partir du même point de référence qui est le point le plus haut à gauche de la première partition décrite. Ceci facilite énormément la tâche de positionnement de régions dans l'image globale.

a. Reconstruction des contours élémentaires:

Le traitement de chaque contour élémentaire commence par l'extraction des informations le concernant à partir du descripteur, ensuite le calcul des éléments de ses deux extrémités ainsi que le centre du cercle passant par les deux extrémités dans le cas d'un contour courbé.

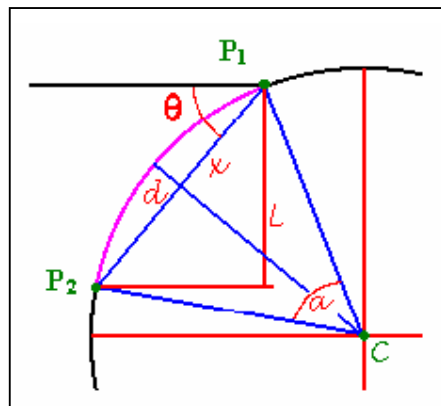


Figure 3.9 : Eléments d'un Contour

Un contour élémentaire est traité soit du haut en bas ou inversement. Ceci est effectué comme suit:

- Initialiser le premier point d'extrémité (point du départ ou point d'arrivée selon le sens du traitement) directement en fonction de la position de celui-ci dans la silhouette
- Calculer le point de l'autre extrémité en prenant en considération les coordonnées du premier point et les informations fournies dans le descripteur (longueur 'L' et angle d'inclinaison 'θ' voir figure 3.9).
- Dans le cas d'un contour courbé, celui-ci est considéré comme étant circulaire (arc circulaire), ainsi le calcul du point du centre du cercle qui passe par les extrémités de ce contour permet de dessiner celui-ci.

Pour pouvoir calculer le centre du cercle qui passe par les deux points d'extrémité il faut tout d'abord connaître l'angle (α) formé au centre par l'intersection des deux rayons (voir la figure 3.9), ainsi que le rayon du cercle.

A partir de la figure 3.9 on peut déduire les relations suivantes:

$$\begin{aligned} \sin(\alpha/2) &= \frac{1}{2}X/r & \implies & d/X = (1 - \cos(\alpha/2)) / 2 \sin(\alpha/2) \text{ ----- (1)} \\ \cos(\alpha/2) &= (r-d)/r \\ & (r \text{ est le rayon du cercle}) \end{aligned}$$

$$\sin^2(\alpha/2) + \cos^2(\alpha/2) = 1 \text{ ----- (2)}$$

La solution de ces deux équations (1) et (2) donne :

$$\begin{aligned} \alpha &= 2 \arcsin(4(d/X)/(4(d/X)^2+1)) \\ r &= L(4(d/X)^2+1)/(8(d/X)\sin(\theta)) \end{aligned}$$

$$\begin{aligned} \text{Sachant que } X &= L/\sin(\theta) \text{ si } \theta \text{ est différent de } 0 \text{ et } 180 \\ X &= L \quad \text{sinon} \end{aligned}$$

Ainsi selon le cas du type du contour (convexe ou concave) et de la frontière dont il fait partie (gauche ou droite), on peut distinguer les deux cas suivants:

- Dans le cas d'un contour convexe qui fait partie de la frontière gauche ou bien d'un contour concave qui fait partie de la frontière droite, le centre du cercle $C(X_c, Y_c)$ est à droite des deux points $P_1(X_1, Y_1)$ et $P_2(X_2, Y_2)$ dont ses coordonnées sont reconnues par :

$$\begin{aligned} X_c &= X_1 + r * \sin(d/x - \frac{1}{2}\alpha) \\ Y_c &= Y_1 + r * \cos(d/x - \frac{1}{2}\alpha) \end{aligned}$$

- Dans les deux cas contraires le centre est à gauche des deux points P_1 et P_2 et est identifié par:

$$\begin{aligned} X_c &= X_1 - r * \sin(d/x + \frac{1}{2}\alpha) \\ Y_c &= Y_1 - r * \cos(d/x + \frac{1}{2}\alpha) \end{aligned}$$

La figure 3.10 illustre ces deux cas:

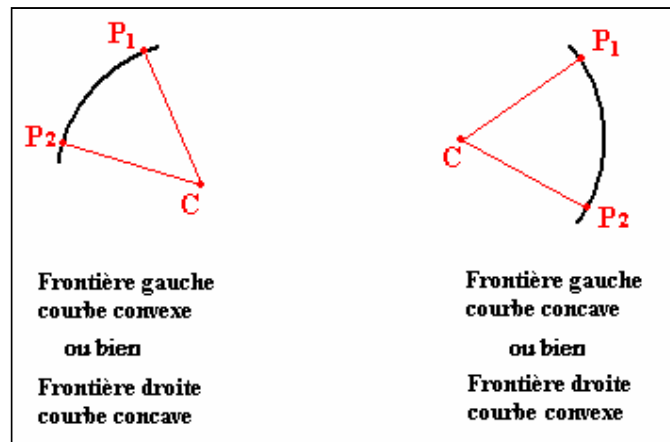


Figure 3.10 : Calcul du centre de la courbe passant par le contour

b. Reconstruction des partitions:

Une partition est reconstruite soit du haut en bas ou inversement selon sa position dans la partie composée (en haut ou en bas de la ligne de jonction) et selon le sens dans lequel la partie composée est reconstruite.

Sa reconstruction s'effectue toujours par rapport à deux points de référence qui sont soit les deux points du démarrage de ses deux frontières gauche et droite ou bien leurs points de terminaison [DAH 06].

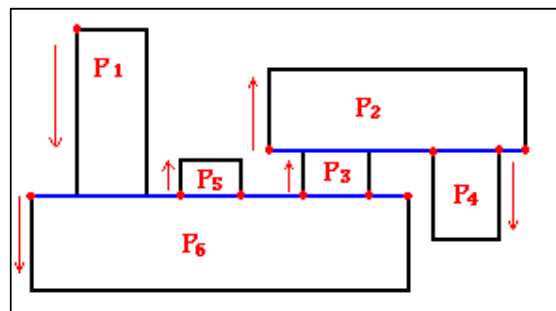


Figure 3.11 : Sens de reconstruction d'une partition

La figure 3.11 montre le sens de reconstruction des partitions ainsi que les points de référence de chacune.

➤ **Reconstruction du haut en bas:**

Une partition est reconstruite du haut en bas lorsque:

- La silhouette est constituée d'une seule partition
- Il s'agit de la première partition décrite dans la silhouette (c'est le cas de la partition P₁ dans la figure 3.11).
- La partition est en bas de la ligne de jonction dans une partie composée traitée du haut en bas (le cas de la partition P₆).

- La partition est en bas d'une ligne de jonction dans une partie composée traitée du bas en haut, sauf pour celle qui est considérée comme référence dans la partie composée (le cas de la partition P_4).

Dans ces cas, les contours élémentaires dans les deux frontières sont reconstruits en commençant par le premier contour pour lequel la première extrémité est considérée comme étant le point de référence P_{t1} et son point d'arrivée est calculé en fonction de P_{t1} et des informations extraites du descripteur. Les contours qui suivent chacun d'eux prend le point d'arrivée du contour précédent comme point du départ et calcule son point d'arrivée de la même manière.

➤ **Reconstruction du bas vers le haut:**

La reconstruction d'une partition est faite du bas vers le haut dans les cas suivants:

- Les partitions qui sont en haut d'une ligne de jonction dans une partie composée traitée du bas en haut (le cas de la partition P_2).
- Les partitions qui sont en haut d'une ligne de jonction dans une partie composée traitée du haut vers le bas et qui ne sont pas considérées comme partitions de référence (cas de la partition P_5).
- La partition de référence dans une partie composée traitée du bas en haut: qui est la partition qui lie les deux parties composées (celle déjà reconstruite et celle qui est en cours) (le cas de la partition P_3).

Dans ce cas l'ordre du traitement des contours élémentaires dans les deux frontières est inversé. Le point de départ du dernier contour dans la frontière gauche est initialisé par le point d'arrivée du premier segment dans la ligne de jonction lié avec la partition en cours de construction, et le point d'arrivée du dernier contour dans la frontière droite, est initialisé par l'autre extrémité du segment dans la ligne de jonction lié avec cette partition. Leurs points du départ sont ainsi calculés en fonction de ces deux points.

Pour le reste des contours chacun d'eux initialise son point de départ par le point du départ du contour élémentaire suivant et calcule son point du départ en fonction de celui-ci.

L'algorithme de reconstruction de partition a comme arguments d'entrée les deux points de référence et le sens de reconstruction de la partition, celui-ci est décrit ci-dessous:

Algorithme Reconstruction_Partition (P_{ref1} , P_{ref2} , Sens)

Déclaration:

P_{deb}, P_{fin} : Point

C_i : contour élémentaire

N, M : entier // correspondent respectivement aux nombres de contours élémentaires des frontières gauche et droite.

Début

Si le sens est du haut vers le bas alors

$P_{deb} \leftarrow P_{ref1}$

Pour i de 1 à N faire

Reconstruire_Contour(P_{deb} , Haut_Bas)

P_{deb} ← le point de terminaison de C_i

Fin pour

P_{deb} ← P_{ref2}

Pour i de 1 à M faire

Reconstruire_Contour(P_{deb}, Haut_Bas)

P_{deb} ← le point de terminaison de C_i

Fin pour

Sinon // sens= bas vers le haut

P_{fin} ← P_{ref1}

Pour i de N à 1 faire

Reconstruire_Contour(P_{fin}, Bas_Haut)

P_{fin} ← le point de début de C_i

Fin pour

P_{fin} ← P_{ref2}

Pour i de M à 1 faire

Reconstruire_Contour(P_{fin}, Bas_Haut)

P_{fin} ← le point de début de C_i

Fin pour

Fin Si

Fin.

c. Reconstruction des lignes de jonctions:

Les informations concernant chaque segment des lignes de jonction sont extraites et sauvegardées. Ces informations sont le type du segment ('s', 'w', 'h'), les partitions liées à celui-ci et sa longueur. Ces informations permettent d'une part de calculer le point du départ et le point de terminaison du segment et de l'autre part de positionner les différentes partitions liées à celui-ci.

Le **segment de référence** est le premier segment lié à la partition déjà reconstruite, celui-ci est le premier à reconstruire dans toute la ligne de jonction. Les segments qui sont à gauche du segment de référence sont reconstruits de droite à gauche dans l'ordre inverse de leur apparition dans la description, tandis que ceux qui sont à droite de celui-ci (y compris ceux qui peuvent être liés avec la partition déjà reconstruite) sont reconstruits de gauche à droite dans l'ordre exact de leur apparition dans la description.

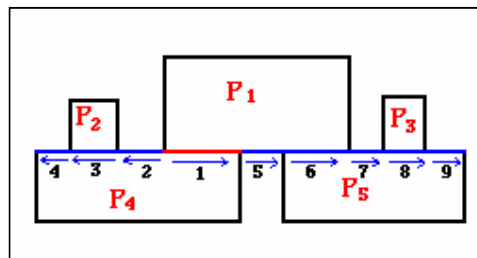


Figure 3.12: Sens et ordre de reconstruction des segments

La figure 3.12 illustre sur un exemple, l'ordre et le sens de reconstruction des segments dans une ligne de jonction. Le segment de référence est coloré en rouge, après la reconstruction de celui-ci les segments 2, 3, et 4 sont reconstruits dans cet ordre et de droite à gauche. Ensuite les segments à droite du segment de référence sont

reconstruits de gauche à droite dans l'ordre exact de leur apparition dans la ligne de jonction.

La reconstruction d'un segment consiste à:

- Initialiser la première extrémité
- Calculer sa seconde extrémité connaissant la première extrémité et la longueur du segment.

L'algorithme de reconstruction de la ligne de jonction utilise la partition déjà reconstruite afin de propager la reconstruction de tous éléments de la silhouette.

➤ **Algorithme de reconstruction :**

L'algorithme de reconstruction d'une ligne de jonction est le suivant :

Algorithme Reconstruire_Ligne_Jonction (P_{ref})

// P_{ref} est la partition de référence (déjà reconstruite)

Déclaration:

$S_1, S_2, \dots, S_n$: Segments

$P_{d1}, P_{d2}, \dots, P_{dn}, P_{f1}, P_{f2}, \dots, P_{fn}$: Points // P_{di} point de départ du segment S_i , et P_{fi} son point d'arrivée

Début:

Rechercher le segment de référence S_i (le 1^{er} segment lié avec P_{ref})

Si P_{ref} est en haut alors

$P_{di} \leftarrow$ point de terminaison de la frontière gauche de P_{ref}

Calculer(P_{fi})

Sinon

$P_{di} \leftarrow$ point de début de la frontière gauche de P_{ref}

Calculer(P_{fi})

Fin si

Pour j de i-1 à 1 faire

$P_{fj} \leftarrow P_{dj+1}$

Calculer(P_{dj})

Fin pour

Pour j de i+1 à n faire // n le nombre de segments

$P_{dj} \leftarrow P_{fj+1}$

Calculer(P_{fj})

Fin pour

Fin.

➤ **Explication :**

Soit la ligne de jonction J contenant les segments suivants : $\{S_1, S_2, \dots, S_n\}$. On va étudier le comportement de cet algorithme à travers les cas suivants:

- Cas 1: le segment de référence S_i est le premier segment dans la ligne de jonction ($i=1$).
- Cas 2: le segment de référence S_i est le dernier segment dans la ligne de jonction ($i=n$).

- Cas général: le segment de référence S_i à des segments à gauche et des segment à droite ($1 < i < n$)

Cas1:

S_1 est le segment de référence, et doit être reconstruit le premier de gauche à droite, et ceci en initialisant son point du début à partir du point de liaison avec la partition de référence, et en calculant ensuite son point d'arrivée. Ensuite on passe directement à la reconstruction des segments qui sont à droite, ceux-ci sont reconstruits de gauche à droite et dans l'ordre: $S_2, S_3, \dots, S_n$.

Cas2:

S_n est le segment de référence, et doit être reconstruit de la même façon que dans le cas précédent, ensuite on passe à la reconstruction des segments qui sont à gauche, et ce en les traitant de droite à gauche en commençant par le segment plus proche du S_n . Donc ceux-ci sont reconstruits dans cet ordre: $S_{n-1}, S_{n-2}, \dots, S_1$.

Cas général:

S_i est le segment de référence, et doit être reconstruit de la même façon que dans les cas précédents, après la reconstruction de celui-ci les segments à gauche sont reconstruits de la même façon que dans le cas précédent $S_{i-1}, S_{i-2}, \dots, S_1$, ensuite les segments à droite sont reconstruit de la même façon que dans le 1^{er} cas : $S_{i+1}, S_{i+2}, \dots, S_n$.

Donc l'algorithme peut reconstruire n'importe quelle ligne de jonction, quelque soit sa structure.

d. Reconstruction des parties composées:

Une partie composée peut contenir des partitions ou des parties composées en plus des lignes de jonction ou disjonction. Les éléments constituant une partie composée sont reconstruits dans les deux sens (du haut vers le bas ou du bas vers le haut).

La reconstruction d'une partie composée commence toujours par la reconstruction de son élément de référence qui est :

- Dans le cas de la partie composée globale, c'est la partition ou la partie composée qui apparaît en premier lieu dans le descripteur,

- Dans le cas d'une partie composée qui fait partie d'une autre partie composée, cet élément est la partition ou la partie composée liée avec la ligne de jonction déjà reconstruite.

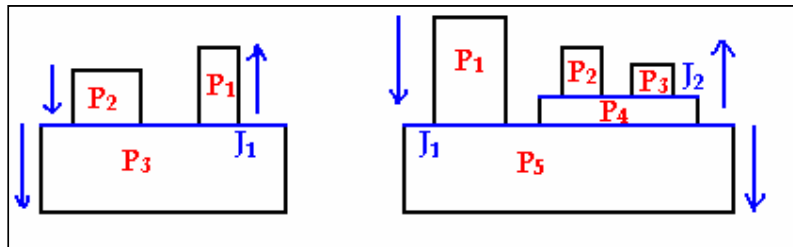


Figure 3.13 : Reconstruction d'une partie composée.

Dans l'image à gauche (figure 3.13), la 1^{ère} partition décrite est P_2 , celle-ci est l'élément de référence de la partie composée.

Dans l'image à droite la partition P_1 est l'élément de référence de la partie composée globale et P_4 est l'élément de référence de la partie composée ($P_2P_3J_2P_4$).

Par convention on dit qu'une partie composée est reconstruite du haut vers le bas si son élément de référence est en haut de la ligne de jonction, et du bas vers le haut dans le cas contraire.

Après la reconstruction de l'élément de référence, la ligne de jonction est positionnée par rapport à celui-ci.

Ensuite les partitions et parties composées en liaison avec cette ligne de jonction, seront reconstruites du bas vers le haut pour celles qui sont en haut et du haut vers le bas pour celles qui sont en bas de la ligne de jonction.

➤ **Algorithme de reconstruction :**

L'algorithme de reconstruction de partie composée est détaillé ci-dessous, il a comme arguments le pointeur vers la partie composée à reconstruire, deux points de référence à partir desquels la reconstruction commence et le sens de reconstruction. Les deux points de référence d'une partie composée sont les deux points de liaison avec la partie déjà reconstruite. Les deux points de référence de la partie composée globale (silhouette) sont considérés comme l'origine du repère (initialisés à (0,0)).

Algorithme Reconstruire_Partie_Composée (PC, Pt₁,Pt₂, Sens)

(Pt₁, Pt₂ : points de référence)

Début:

Si l'élément de référence est une partition (P_{ref}) alors

Reconstruire_Partition P_{ref} (Pt₁, Pt₂, Sens)

Reconstruire_Ligne_Jonction(P_{ref})

Pour toute partition (P_i≠P_{ref}) en haut de la ligne de jonction faire

Calculer les points de liaison Pt_{i1}et Pt_{i2} de P_i avec la ligne de jonction

Reconstruire_Partition P_i (Pt_{i1}, Pt_{i2}, bas_vers_haut)

Fin pour

Pour toute partie composée (PC_i) en haut de la ligne de jonction faire

Calculer les points de liaison Pt_{i1} et Pt_{i2} de PC_i avec la ligne de jonction

Reconstruire_Partie_Composée(PC_i, Pt_{i1}, Pt_{i2}, bas_vers_haut)

Fin pour

Pour toute partition (P_i≠P_{ref}) en bas de la ligne de jonction faire

Calculer les points de liaison Pt_{i1} et Pt_{i2} de P_i avec la ligne de jonction

Reconstruire_Partition(Pt_{i1}, Pt_{i2}, haut_vers_bas)

Fin pour

Pour toute partie composée (PC_i) en bas de la ligne de jonction faire

Calculer les points de liaison Pt_{i1}et Pt_{i2} de PC_i avec la ligne de jonction

Reconstruire_Partie_Composée(PC_i, Pt_{i1}, Pt_{i2}, haut_vers_bas)

Fin pour

Sinon // l'élément de référence est une partie composée

Reconstruire_Partie_Composée (PC_{ref} , P_{t1} , P_{t2} , Sens)

Rechercher la partition P_{ref} dans PC_{ref} lié avec la ligne de jonction J_n

Reconstruire_Ligne_Jonction (P_{ref})

Pour toute partition (P_i) en haut de la ligne de jointure faire

Calculer les points P_{t1} et P_{t2} de liaison de P_i avec J_n

Reconstruire_Partition P_i (P_{t1} , P_{t2} , bas_en_haut)

Fin pour

Pour toute partie composée ($PC_i \neq PC_{ref}$) en haut de la ligne de jointure faire

Calculer les points de liaison P_{t1} et P_{t2} de PC_i avec J_n

Reconstruire_Partie_Composée(PC_i , P_{t1} , P_{t2} , bas_en_haut)

Fin pour

Pour toute partition (P_i) en bas de la ligne de jointure faire

Calculer les points de liaison P_{t1} et P_{t2} de P_i avec J_n

Reconstruire_Partition P_i (P_{t1} , P_{t2} , haut_en_bas)

Fin pour

Pour toute partie composée ($PC_i \neq PC_{ref}$) en bas de la ligne de jointure faire

Calculer les points de liaison P_{t1} et P_{t2} de PC_i avec J_n

Reconstruire_Partie_Composée(PC_i , P_{t1} , P_{t2} , haut_en_bas)

Fin pour

Fin si

Fin.

A chaque rencontre d'un élément de type partie composée le processus est lancé d'une manière récursive.

➤ **Explication :**

Soit PC une partie composée de structure quelconque. Le processus de reconstruction commence sa reconstruction du haut en bas en cherchant tout d'abord son élément de référence.

Si l'élément de référence est une partition, celui-ci fait appel à la reconstruction de partition. Après la reconstruction de la 1^{ère} partition, la reconstruction de ligne de jonction est lancée en faisant appel au processus de reconstruction de cette dernière. Celui-ci fait une recherche sur les segments à l'intérieur de la ligne de jonction pour repérer celui lié avec la partition reconstruite, qui sera positionné à l'endroit exact par rapport à cette partition. Ensuite tout segment à gauche et à droite du segment de référence sera placé à sa position. Dans la cas où l'élément de référence est une partie composée le même processus sera lancé d'une manière récursive pour la reconstruction de celui-ci.

Après la reconstruction de la ligne de jonction le processus de reconstruction de partition est lancé pour reconstruire toute partition au dessus de la ligne de jonction du bas en haut. Chaque processus doit rechercher dans la ligne de jonction les points de liaison avec la partition cible, ces points seront considérés comme points de référence.

Ensuite pour chaque partie composée au dessus de la ligne de jonction le processus de reconstruction doit reconstruire celles-ci du bas en haut, en cherchant tout d'abord l'élément de référence de chacune, cet élément est reconstruit du bas en haut. Ensuite le positionnement de ligne de jonction par rapport à l'élément de référence et enfin la reconstruction des éléments qui restent de la même manière.

Ces deux dernières opérations seront répétées pour la reconstruction des éléments en bas de la ligne de jonction mais dans le sens du haut en bas.

Exemple:

Examinons le déroulement de cet algorithme sur l'exemple de la figure 3.14.

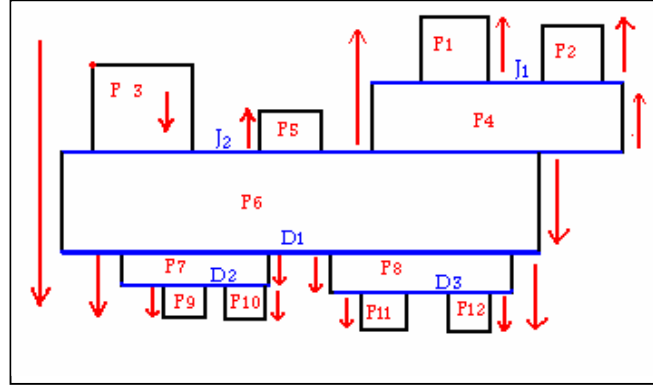


Figure 3.14: Exemple de reconstruction d'une partie composée

La partie composée globale est $((P_3P_5(P_1P_2J_1P_4))J_2P_6)D_1(P_7D_2P_9P_{10})(P_8D_3P_{11}P_{12})$.

L'élément de référence de toute la partie composée est la partie composée $P_3P_5(P_1P_2J_1P_4))J_2P_6$, celui-ci est le premier élément à construire (de haut en bas), cependant étant donné que cet élément est une partie composée sa construction est faite par un appel récursif au même processus.

L'élément de référence dans la partie composée $P_3P_5(P_1P_2J_1P_4))J_2P_6$ est P_3 qui est une partition, le processus de reconstruction de partition est appelé ainsi afin de construire celui-ci (de haut en bas).

Après la reconstruction de P_3 la ligne jonction J_2 est positionnée selon ses points de liaison avec P_3 . Ensuite la partition P_5 et la partie composée $P_1P_2J_1P_4$ sont reconstruites de bas en haut.

La reconstruction de la partie composée $P_1P_2J_1P_4$ se déroule comme suit : la reconstruction de la partition P_4 (de bas en haut) en premier lieu, deuxièmement le positionnement de la ligne de jonction J_1 et en fin la reconstruction des partitions P_1 et P_2 (du bas en haut).

Ensuite la partition P_6 est ainsi reconstruite du haut en bas.

Après la reconstruction de $P_3P_5(P_1P_2J_1P_4))J_2P_6$ le contrôle retourne au premier processus qui continue son exécution par le positionnement de la ligne de disjonction D_1 , ensuite les deux parties composées $P_7D_2P_9P_{10}$ et $P_8D_3P_{11}P_{12}$ sont reconstruites du haut en bas en suivant le même procédé.

Ainsi l'ordre globale dans lequel les partitions et les lignes de jointure sont reconstruites est: $P_3, J_2, P_5, P_4, J_1, P_1, P_2, P_6, D_1, P_7, D_2, P_9, P_{10}, P_8, D_3, P_{11}, P_{12}$.

2) Reconstruction de région:

Une région est constituée d'une silhouette avec un fond coloré et d'un ensemble de régions internes (voir la figure 3.15). Ainsi reconstruire une région revient à reconstruire son contour externe en prenant en considération sa position relativement à l'image globale, sa couleur, et ses régions internes.

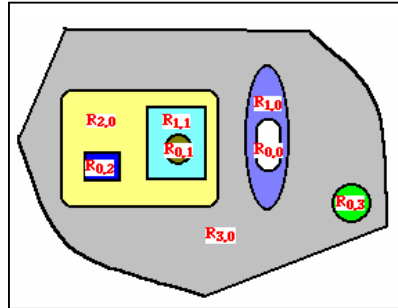


Figure 3.15: Structure d'une région

a. Positionnement de région:

La reconstruction de silhouette se base sur le calcul des différents points de celle-ci par rapport au repère placé sur le point de référence de la silhouette.

Or la position de région donnée dans le descripteur LWDI correspond à la position du point le plus haut à gauche de la silhouette. Cette position est donnée par rapport au repère placé au coin le plus haut à gauche de l'image globale.

Le positionnement consiste donc à calculer la position du point le plus haut à gauche dans la silhouette par rapport au point de référence, et déduire le décalage qui existe entre cette position et la position donnée dans le descripteur. Ensuite ce décalage est appliqué sur l'ensemble des points calculés dans toute la silhouette. En effet ce décalage correspond à la position du point de référence par rapport à l'image globale (voir la figure 3.16).

❖ **Calcul du point le plus haut à gauche dans la silhouette par rapport au point de référence:**

La description d'une partition commence toujours à partir du point le plus haut à gauche de celle-ci, ce point correspond au point du début de la frontière gauche.

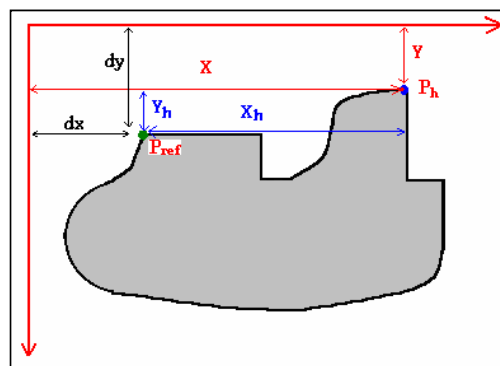


Figure 3.16 : Positionnement d'une région

Dans le cas où la silhouette est constituée d'une seule partition, le point le plus haut à gauche de cette silhouette est obtenu directement à partir de la partition (le même point que celui de la partition). Par contre dans le cas où la silhouette est constituée de plusieurs partitions, le point le plus haut à gauche dans la silhouette est obtenu en appliquant une comparaison sur l'ensemble des points les plus haut à gauche de toutes les partitions, et choisir celui le plus haut parmi ces points. Soient :

- P_{ref} le point de référence de la région, x_{ref} et y_{ref} ses coordonnées par rapport à l'image globale.

- P_h le point le plus haut à gauche de la silhouette, X_h et Y_h ses coordonnées par rapport au points de référence (voir la figure 3.16).

- (X, Y) la position de la région donnée par le descripteur qui correspond à celle du point le plus haut à gauche P_h .

Ainsi le décalage à faire pour obtenir ces nouvelles coordonnées correspond à $dx = X - X_h$ sur l'axe des abscisses et $dy = Y - Y_h$ sur celui des ordonnées.

Ce décalage correspond à la position du point de référence par rapport à l'image globale $P_{ref}(X_{ref}=dx, Y_{ref}=dy)$ (voir la figure précédente).

Donc le positionnement consiste à appliquer le décalage $(X-dx, Y-dy)$ sur tout point calculé dans la silhouette.

b. Colorisation de région:

La colorisation consiste à remplir l'ensemble des éléments constituant celle-ci (partitions, lignes de jonction, parties composées) par la couleur décrite dans le descripteur.

❖ Colorisation de partition:

La colorisation de partition consiste à appliquer un balayage tout au long de ses deux frontières du haut en bas, et à colorier les points entre ces deux frontières.

Ce procédé commence tout d'abord par le positionnement de deux points variables Pt_1 et Pt_2 sur les deux points de début des deux frontières, et à tracer une ligne droite horizontale entre ces derniers.

Le balayage des deux frontières est effectué en décalant les deux points Pt_1 et Pt_2 d'une ligne en bas tout en positionnant chacun d'eux sur un point de la frontière correspondante. Cette opération est répétée jusqu'à arriver aux deux points de terminaison des deux frontières.

❖ Colorisation de ligne de jonction:

La colorisation d'une ligne de jonction consiste à dessiner chaque segment faisant partie de celle-ci.

❖ Colorisation de partie composée:

La colorisation de partie composée consiste à colorier l'ensemble de partitions et lignes de jonction composant celle-ci.

c. Régions internes :

Sachant qu'une région interne a la même structure qu'une région, celle-ci donc est reconstruite suivant le même principe.

Cependant si dans la reconstruction d'une région on commence par la reconstruction des régions internes avant la région globale, la colorisation du fond de cette dernière risque de détruire les régions internes déjà reconstruites.

Ce problème est résolu en commençant par la colorisation des régions les plus externes.

Exemple :

La colorisation de l'image dans la figure 3.17 sera effectuée dans l'ordre suivant :

$R_{3,0}, R_{1,0}, R_{0,0}, R_{2,0}, R_{1,1}, R_{0,1}, R_{0,2}, R_{0,3}$.

L'algorithme de reconstruction de région sera comme suit :

Algorithme Reconstruire_Région (R, Couleur, Position)

Déclaration :

R : Ensemble de région interne R_i (R peut être vide)

Début :

Reconstruire_Silhouette (Descripteur DXLWDOS)

Positionner_Silhouette(*Position*)

Colorier_Silhouette (Couleur)

Pour toute région $R_i \in R$ *faire*

Reconstruire_Région (R_i , couleur de R_i , position de R_i)

Fin pour

Fin

d. Explication :

Une région peut comporter un nombre N ($N \geq 0$) de régions internes, et dont chaque région interne peut contenir à son tour un nombre M ($M \geq 0$) de régions internes et ainsi de suite.

L'algorithme de reconstruction doit donc être capable de reconstruire l'ensemble de régions qui existent que se soit en allant en largeur dans une même région ou en allant en profondeur d'une région à une région interne.

❖ **Balayage en largeur :**

Soit une région $R_{i,j}$ contenant les régions internes $R_{I1,0}, R_{I2,1}, \dots, R_{In,n}$. ($I_1, I_2, \dots, I_n$ sont inférieure à i). La reconstruction de $R_{i,j}$ commence par la reconstruction de sa silhouette, ensuite son positionnement et sa colorisation.

Pour la reconstruction des régions internes la boucle s'occupe de cette tâche, ainsi le processus de reconstruction de région sera lancer N fois ($N \geq 0$: le nombre de régions internes) pour reconstruire l'ensemble de ces régions.

❖ **Balayage en profondeur :**

Soit une région $R_{i,j}$ contenant au moins une région interne $R_{I1,J1}$, et dont cette dernière contient au minimum une région interne $R_{I2,J2}$, et ainsi de suite pour n'importe quelle région interne.

La reconstruction de $R_{i,j}$ se caractérise donc par un appel à la reconstruction de la silhouette de la région globale ensuite son positionnement et sa colorisation.

La reconstruction de la région interne s'effectue par un appel récursif au même processus de reconstruction, qui à son tour après la reconstruction de la silhouette, son positionnement et sa colorisation fait un appel récursif à lui-même pour la reconstruction de la 1^{ère} région interne, et ainsi de suite pour toute région contenant une région interne.

Donc l'algorithme est capable de reconstruire l'ensemble de régions internes qui existent que se soit en allant en largeur ou bien en profondeur dans une région.

3) Reconstruction de l'image:

L'image est constituée d'un fond et d'une suite de régions superposées sur ce fond. Donc la reconstruction de l'image est la reconstruction de l'ensemble de ses éléments.

Pour la même raison citée dans l'ordre de reconstruction des régions, le fond de l'image (arrière plan) doit être reconstruit avant toute région dans l'image, par contre l'ordre de reconstruction des régions superposées directement dans l'arrière plan de l'image n'est pas important. Celles-ci sont reconstruites dans leur ordre d'apparition dans le descripteur.

Le processus de reconstruction de l'image se résume dans:

- La reconstruction de l'arrière plan (dimensions et couleur du fond).
- La reconstruction de l'ensemble de régions.

Algorithme Reconstruire_Image (Descripteur DXLWDI)

Déclaration :

R : Ensemble de région R_i dans l'image (R peut être vide)

Début :

Reconstruire_Arrière_Plan (dimensions, couleur du fond)

Pour toute région $R_i \in R$ faire

Reconstruire_Région (R_i , couleur de R_i , position de R_i)

Fin pour

Fin

La figure 3.17 illustre l'ordre de reconstruction des composants de l'image.

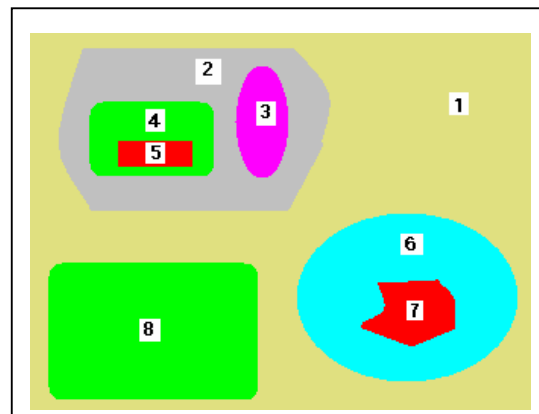


Figure 3.17 : Reconstruction d'image

III.3. Démonstration:

Pour démontrer la validité de cet algorithme environs toute structure d'image, nous allons utiliser la démonstration par récurrence.

Tout d'abord nous allons étudier le comportement de cet algorithme pour des cas simples :

- **Cas 1:** image ne contenant aucune région.
- **Cas 2 :** image contenant une ou plusieurs régions de niveau 0.
- **Cas 3 :** image contenant au moins une région de niveau égal à 1.

Pour schématiser le comportement de l'algorithme et faciliter sa compréhension on désigne par :

Proc_Image : le processus de reconstruction d'image.

Proc_Reg(R_i): le processus de reconstruction de région R_i .

Proc_Sil(R_i) : l'ensemble des processus de reconstruction, positionnement, et colorisation du fond de la silhouette de la région R_i .

Cas1 : Image ne contenant aucune région ($R=\emptyset$) :

La reconstruction de l'image se limite uniquement dans la reconstruction de l'arrière plan.

Donc après l'appel du processus de reconstruction de l'arrière plan (dimensions et couleur) l'exécution du processus appelant se termine sans entrer dans la boucle qui traite les régions. Ce comportement est schématisé ci-dessous:

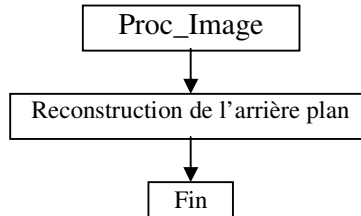


Figure 3.18: Schéma d'exécution (Cas1)

Résultat : L'image donc est reconstruite correctement.

Cas 2 : ($R= \{R_{0,0}, R_{0,1}, ..., R_{0,n}\}$) :

Après le traitement de l'arrière plan le processus de reconstruction d'image entre dans l'exécution de la boucle qui traite les régions, ainsi le processus de reconstruction de région est lancé sur la 1^{ère} région $R_{0,0}$ ($Proc_Reg(R_{0,0})$), ce dernier va appeler les processus de reconstruction, positionnement, et colorisation de la silhouette de cette région ($Proc_Sil(R_{0,0})$). Après l'exécution de ces derniers le contrôle retourne au processus appelant ($Proc_Reg(R_{0,0})$).

Vu que cette région ne contient pas de régions internes ce dernier ne va pas traiter la boucle qui traite les régions internes, ainsi le contrôle revient au premier processus ($Proc_Image$).

Celui-ci qui est en cours d'exécution de la boucle va reconstruire la 2^{ème} région ($R_{0,1}$).

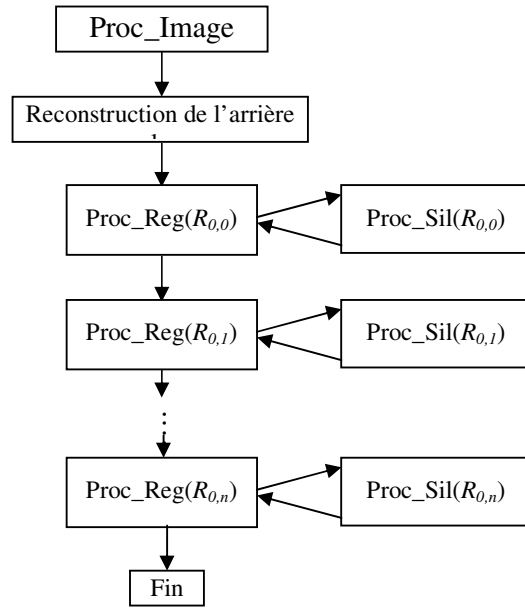


Figure 3.19: Schéma d'exécution (Cas2)

Les mêmes opérations seront répétées jusqu'à la reconstruction de la dernière région ($R_{0,n}$). Le comportement de l'algorithme est schématisé dans la figure 3.19.

Résultat : L'image donc est reconstruite correctement.

Cas3 : ($R = \{R_{l,0}, R_{l,1}, \dots, R_{l,n}\}$ et $l \in \{0, 1\}$) :

Le processus *Proc_Image* commence tout d'abord par l'appel du processus de reconstruction de l'arrière plan, ensuite il entre dans la boucle qui traite les régions.

Donc le processus *Proc_Reg* sera appelé pour la reconstruction de la 1^{ère} région $R_{l,0}$ (*Proc_Reg*($R_{l,0}$)) ; celui-ci va demander l'exécution des processus *Proc_Sil* sur cette région (*Proc_Sil*($R_{l,0}$)).

Après l'exécution des processus *Proc_Sil* ($R_{l,0}$) le contrôle retourne au processus appelant *Proc_Reg*($R_{l,0}$). Comme la région $R_{l,0}$ contient des régions internes ($l=1$), ce processus va entrer dans l'exécution de la boucle qui traite les régions internes. Vu que les régions internes sont traitées par le même processus, ce dernier va marquer un point d'arrêt à ce niveau en sauvegardant son état d'exécution, et faire un appel d'exécution récursive à lui-même afin de reconstruire la 1^{ère} région interne $R_{0,0}$ (*Proc_Reg*($R_{0,0}$)).

Le processus *Proc_Reg*($R_{0,0}$) va demander l'exécution du processus *Proc_Sil*($R_{0,0}$), ensuite il retourne le contrôle au processus appelant (*Proc_Reg*($R_{l,0}$)) sans entrer dans la boucle puisque la région interne $R_{0,0}$ n'a pas de régions internes.

De nouveau le processus *Proc_Reg*($R_{l,0}$) en cours d'exécution d'une boucle, va marquer un point d'arrêt en sauvegardant son état d'exécution, et faire un appel d'exécution récursive à lui-même afin de reconstruire la 2^{ème} région interne $R_{0,1}$ (*Proc_Reg*($R_{0,1}$)). Ces opérations seront répétées jusqu'à la dernière région interne $R_{0,m}$.

Après la reconstruction de la dernière région interne dans la région $R_{l,0}$ le processus *Proc_Reg*($R_{l,0}$) se termine en retournant le contrôle au processus racine

(Proc_Image), qui va répéter la même opération pour la reconstruction de la 2^{ème} région $R_{l,l}$, et ainsi de suite jusqu'à la reconstruction de la dernière région $R_{l,n}$.

Ce comportement est schématisé ci-dessous :

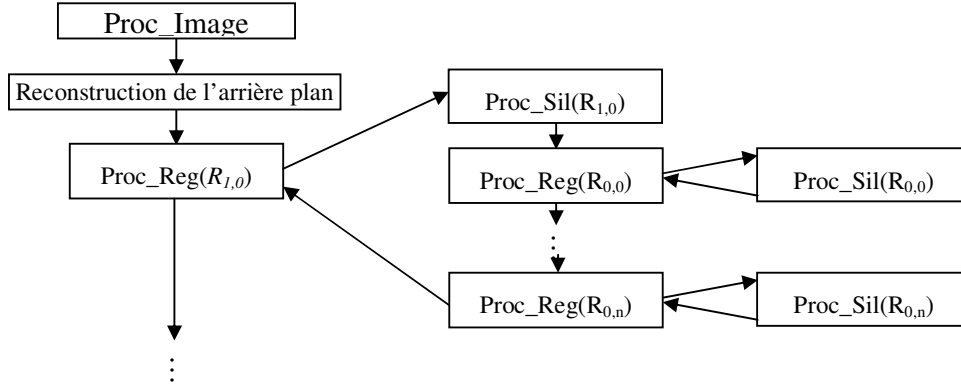


Figure 3.20: Schéma d'exécution (Cas3)

Le comportement de l'algorithme est schématisé dans la figure 3.20.

Résultat : Donc l'image dans ce cas est reconstruite correctement.

Cas général :

On suppose que l'algorithme est valide pour la reconstruction d'une image contenant un ensemble de régions dont au minimum une de ces régions est d'ordre p ($R = \{R_{p,0}, R_{p,1}, \dots, R_{p,n}\}$ avec $p_1, \dots, p_n \in [0..p]$), et on démontre qu'il peut reconstruire une image contenant un ensemble de régions dont au minimum une de ces régions est d'ordre $p+1$ ($R = \{R_{p+1,0}, R_{p+1,1}, \dots, R_{p+1,n}\}$).

L'algorithme peut construire une image contenant un ensemble de régions d'ordre p : ($R = \{R_{p,0}, R_{p,1}, \dots, R_{p,n}\}$), ceci signifie que le processus Proc_Image a fait appel au processus Proc_Reg n fois ($Proc_Reg(R_{p,0}), Proc_Reg(R_{p,1}), \dots, Proc_Reg(R_{p,n})$) et dont chacun de ces processus a fait appel récursif à lui-même respectivement $p, p_1, \dots, p_n$ fois afin de reconstruire ces régions internes.

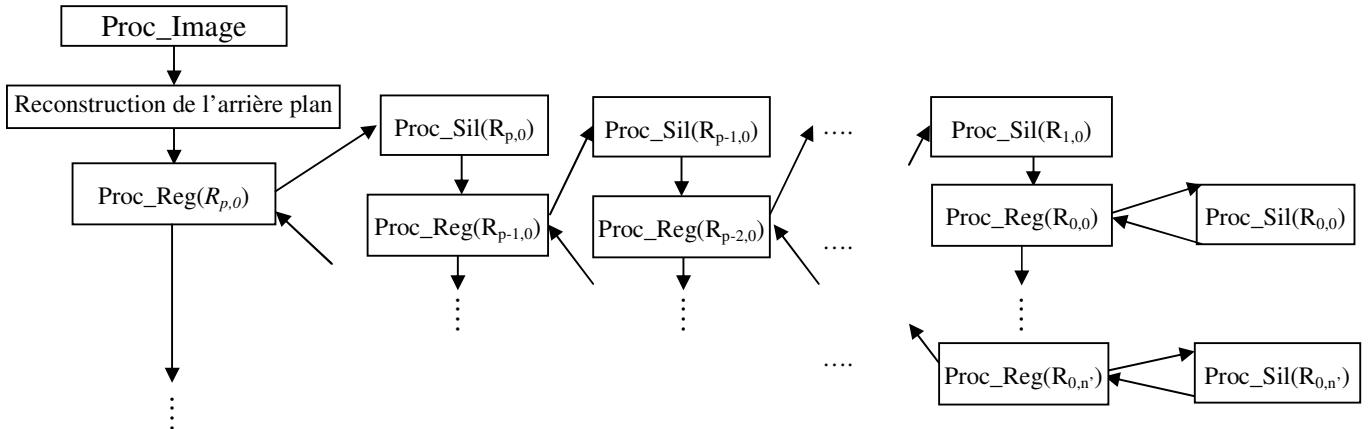


Figure 3.21: Schéma d'exécution (Cas général)

Maintenant supposons que toutes ces régions sont mises dans une nouvelle région R' , ainsi l'ordre de cette région est égal à l'ordre le plus élevé parmi les régions internes à celle-ci plus 1, c'est-à-dire $p+1$.

L'ajout d'une région R' se caractérise dans la reconstruction de l'image par un appel supplémentaire au processus $Proc_Reg(R')$.

Ainsi ce processus va commencer tout d'abord par un appel aux processus de reconstruction, positionnement, dessin et colorisation de silhouette $Proc_Sil(R')$, ensuite il commence la reconstruction des régions internes qui se manifeste par l'appel des processus qu'on supposé déjà avoir faire ($Proc_Reg(R_{p,0})$, $Proc_Reg(R_{p1,1})$, ..., $Proc_Reg(R_{pn,n})$).

Ainsi l'algorithme est capable de reconstruire une image contenant des régions d'ordre $p+1$. Le comportement de l'algorithme est schématisé dans la figure 3.21.

Résultat : Donc l'algorithme est capable de reconstruire n'importe quelle image.

III.4. Exemple de reconstruction:

Pour étudier le comportement de cet algorithme sur un exemple concret, on va prendre l'image de la figure 3.22.

L'arrière plan de cette image a les dimensions [350x180], et est coloré en jaune. Il contient quatre régions $R = \{R_{2,0}, R_{1,1}, R_{1,2}, R_{1,3}\}$.

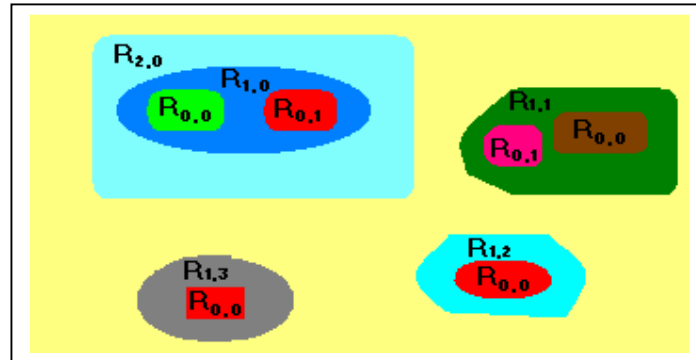


Figure 3.22 : Exemple de reconstruction

- La première région $R_{2,0}$ contient une région interne $R_{1,0}$ qui contient elle aussi deux régions internes $R_{0,0}$, $R_{0,1}$.
- La deuxième région $R_{1,1}$ contient deux régions internes qui sont $R_{0,0}$, $R_{0,1}$.
- La troisième région $R_{1,2}$ et la quatrième $R_{1,3}$ contiennent une seule région interne $R_{0,0}$.

Pour étudier le comportement de cet algorithme sur cette image nous utilisons les mêmes désignations que dans la section précédente ($Proc_Image$, $Proc_Reg()$, $Proc_Sil()$) pour désigner les différents processus de reconstruction.

Étape 0: vérification du descripteur: cette étape ne rentre pas dans la reconstruction de l'image malgré qu'elle soit primordiale, du fait que c'est elle qui décide si l'image peut être reconstruite ou non. Donc si la description de l'image est jugée valide la reconstruction de l'image commence sinon la description doit être corrigée.

Étape1 : Lancement du processus de reconstruction d'image (Proc_Image) : c'est le processus le plus important, son exécution est caractérisée par une suite d'appels aux différents processus descendants. Il commence tout d'abord par la reconstruction de l'arrière plan, ensuite il passe à la reconstruction des régions.

Étape 2 : Reconstruction de l'arrière plan : c'est le premier processus appelé par le processus ancêtre. Celui-ci est chargé de la définition des dimensions de l'arrière plan et de sa colorisation.

Après la fin de ce processus le contrôle retourne au processus appelant (Proc_Image) qui va commencer l'exécution de la boucle.

Étape3 : Exécution de la boucle : ceci va permettre la reconstruction des régions $\{R_{2,0}, R_{1,1}, R_{1,2}, R_{1,3}\}$.

- a. **1^{ère} itération** : reconstruction de la région $R_{2,0}$: ceci est caractérisé par l'appel du processus Proc_Reg($R_{2,0}$).

Le processus Proc_Reg($R_{2,0}$) passe lors de son exécution par l'appel des processus Proc_Sil($R_{2,0}$) qui englobe la reconstruction, le positionnement et la colorisation.

Après l'exécution de ces processus le contrôle revient au processus appelant Proc_Reg($R_{2,0}$) qui va entrer dans la boucle afin de reconstruire les régions internes.

- Exécution de la boucle : cette région ne contient qu'une seule région interne $R_{1,0}$, donc l'exécution de cette boucle va se faire une seule fois :

- ❖ Itération1 : appel récursif du processus Proc_Reg($R_{1,0}$) afin de reconstruire cette région. A son tour ce processus va passer par les étapes suivantes :

- Appel des processus Proc_Sil($R_{1,0}$).

- Exécution de la boucle afin de reconstruire les régions internes $\{R_{0,0}, R_{0,1}\}$. L'exécution de cette boucle se résume dans deux appels récursifs Proc_Reg($R_{0,0}$) et Proc_Reg($R_{0,1}$)

- Itération1 : Appel du processus Proc_Reg($R_{0,0}$) : ce processus va faire appels aux processus Proc_Sil($R_{0,0}$) et rend la main au processus appelant sans exécuter la boucle parce que $R_{0,0}$ ne contient pas de régions internes.

- Itération2 : Appel du processus Proc_Reg($R_{0,1}$) : les mêmes opérations que celle de la région $R_{0,0}$ sont répétées sur celle-ci.

Le processus Proc_Reg($R_{0,1}$) rend le contrôle à son processus appelant Proc_Reg($R_{1,0}$), qui à son tour arrive à sa fin et rend le contrôle à son ancêtre Proc_Reg($R_{2,0}$).

Le processus Proc_Reg($R_{2,0}$) qui s'est arrêté dans l'exécution de la boucle reprend son chemin, mais vu que $R_{2,0}$ ne contient qu'une seule région interne $R_{1,0}$ qui déjà reconstruite celui-ci marque sa fin et rend le contrôle au processus principal Proc_Image qui s'est arrêté à son tour dans l'exécution de la boucle. Donc il termine l'exécution de la 1^{ère} itération (reconstruction de $R_{2,0}$) et va commencer l'exécution de la deuxième.

Le processus $\text{Proc_Reg}(R_{l,l})$ retient le contrôle après la 2^{ème} itération puis termine son exécution en rendant le contrôle au processus principal Proc_Image qui va terminer l'exécution de cette itération et passe à la suivante :

- c. **3^{ème} itération :** reconstruction de la région $R_{l,2}$: (appel du processus $\text{Proc_Reg}(R_{l,2})$). Celui-ci après l'appel de l'ensemble des processus $\text{Proc_Sil}(R_{l,2})$ passe à l'exécution de la boucle afin de reconstruire les régions internes. La région $R_{l,2}$ contient une seule région interne donc la boucle va s'exécuter une seule fois :

- ❖ Itération1 : appel récursif du processus $\text{Proc_Reg}(R_{0,0})$ qui à son tour fait appel aux processus $\text{Proc_Sil}(R_{0,0})$ puis il rend le contrôle au processus appelant.

Le processus $\text{Proc_Reg}(R_{l,2})$ retient le contrôle après la 1^{ère} itération puis termine son exécution en rendant le contrôle au processus principal Proc_Image qui va terminer l'exécution de cette itération et passe à la suivante :

- d. **4^{ème} itération :** reconstruction de la région $R_{l,3}$: (appel du processus $\text{Proc_Reg}(R_{l,3})$). Les mêmes opérations que celle de l'itération précédente sont répétées sur cette région.

Après la reconstruction de la région $R_{l,3}$ toutes les régions sont terminées donc le processus principal Proc_Image termine son travail.

L'exécution de cet algorithme sur cette image est schématisée par la figure 3.23 .

IV. Etude des performances de la méthode

La description d'une image dans le langage LWDI peut avoir un grand intérêt surtout dans les bases de données images du fait qu'elle permet un gain dans l'espace mémoire réservé pour le stockage des images et surtout de la possibilité d'indexer les images afin de pouvoir en appliquer la recherche et la comparaison sémantique [LAR 07].

La possession d'outils permettant la reconstruction d'une image à partir de sa description est d'une grande importance, en raison du besoin à l'information visuelle. En effet la coopération entre ces deux techniques leur permette d'être très utiles surtout dans la transmission sur réseaux.

L'algorithme proposé dans ce chapitre permet donc la reconstruction d'une image à partir de sa description LWDI. Cependant son utilisation reste liée à l'efficacité de celui-ci et à la qualité de l'image produite. Ces deux critères permettent d'évaluer les performances de cet algorithme :

IV.1. Efficacité:

Ce critère signifie la possibilité de reconstituer l'image dans un temps acceptable. En fait les facteurs qui peuvent avoir effet sur le temps de reconstitution sont :

- Le nombre de contours élémentaires dans la silhouette de chaque région.
- Le nombre de régions internes dans l'image.
- Le nombre de points dans chaque région (colorisation).

a. Nombre de contours élémentaires :

Le nombre de contours élémentaires dans une silhouette est égal au nombre de points de courbure détectés tout au long de son contour.

L'algorithme de **Chetverikov** [CHE 03] est utilisé pour la localisation des points de courbure. Cet algorithme utilise deux paramètres pour augmenter ou diminuer le nombre de points de courbure localisés. Ces deux paramètres sont l'angle formé par le contour au niveau du point de test (courbure) et la distance entre ce point et les deux points de contour appartenants aux arêtes de l'angle.

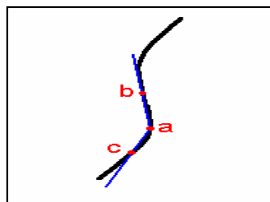


Figure 3.24 : Point de courbure selon l'algorithme de Chetverikov

Le point a est détecté comme point de courbure si l'angle bac est inférieur à un angle donné ' α_c '.

Ainsi l'augmentation de la valeur de α_c permet d'élargir le champ d'angles qui correspondent aux points de courbure, ce qui permet d'augmenter le nombre de points

de courbure, et la diminution de la distance d_{max} ($[ab]$ et $[ac]$) permet de détecter les petites déformations [CHE 03].

Soit N le nombre maximum de points de courbures d'une silhouette. Ce nombre dépend des deux paramètres d_{max} et α_c de l'algorithme cité ci-dessus. En effet le nombre de contours élémentaires augmente avec l'augmentation de la valeur de α_c ou la diminution de la valeur de d_{max} , et il diminue dans les deux cas contraires.

Le temps de reconstruction de la silhouette est la somme des temps mis à la reconstruction des contours élémentaires:

$$T_{sil} = \sum_{i=1}^N T_i$$

T_i est le temps de reconstruction d'un contour élémentaire, il comprend le temps de calcul ainsi que celui du dessin du contour. Le temps T_i dépend du nombre de pixels du contour numéro i .

Soit T_{cmoy} le temps moyen pour la reconstruction d'un contour élémentaire, et N_{cm} le nombre moyen de contours dans une silhouette. Ainsi le temps de reconstruction moyen d'une silhouette est reconnu par $T_{msil} = N_{cm} * T_{cmoy}$.

b. Nombre de points (pixel) par région :

Le temps mis dans la colorisation d'une région dépend du nombre de points M dans celle-ci.

Donc

$$T_{creg} = M * T_{cp} \text{ (} T_{cp} \text{ est le temps colorisation d'un pixel)}$$

Soit N_{mpR} le nombre moyen de pixels dans une région, ainsi le temps de reconstruction moyen d'une région est reconnu par:

$$T_{mreg} = N_{cm} * T_{cmoy} + N_{mpR} * T_{cp}$$

c. Nombre de régions :

Le temps de colorisation de l'arrière plan est la somme du temps de colorisation de chaque point dans celui-ci :

$$T_{cf} = H * L * T_p \text{ (} H \text{:hauteur et } L \text{:largeur de l'image).}$$

Donc le temps de reconstruction de l'image est :

$$T_{img} = H * L * T_p + N_{mR} (N_{cm} * T_{cmoy} + N_{mpR} * T_{cp})$$

Une étude théorique de ce temps de reconstruction montre que :

- Si les régions constituant l'image sont de formes régulières (peu de contours courbés), dans ce cas le temps d'exécution est en fonction linéaire avec le nombre de régions.
- Le temps d'exécution varie toujours linéairement avec le nombre de régions mais il est plus important lorsqu'il y a présence de contours courbés.

IV.2. Qualité de l'image reconstruite

IV.2.1 Introduction :

Ces dernières années, plusieurs avancées ont été réalisées dans le domaine de l'imagerie numérique couleur en ce qui concerne les techniques de restauration d'images et de correction chromatique. Par contre en ce qui concerne l'évaluation de la qualité des images couleur beaucoup reste à faire [OUN 06].

Afin de mesurer la qualité et de pouvoir juger si les images à visualiser sont satisfaisantes en termes de qualité, il est nécessaire de définir clairement des critères de base pour les images numériques "acceptables" et "inacceptables" en fonction de l'application visée.

Si jusqu'à présent la meilleure évaluation reste visuelle (donc subjective) soit par des techniques psychophysiques soit par évaluation experte, il est primordial d'utiliser des critères et des mesures objectifs.

Récemment beaucoup d'études et de publications sont apparues dans le domaine de l'évaluation objective de la qualité des images. Ces travaux sont essentiellement dictés par l'avènement des images et de la vidéo numériques et par les nouveaux besoins en encodage d'images (compression, transmission,...). Par conséquent, la majorité de ces études abordent l'évaluation de la qualité des images d'un angle spécifique à ces domaines en mesurant des défauts spécifiques comme les artefacts de compression (*blocking artefacts* produits par les algorithmes de compression dans les séquences vidéo à faible débit) [OUN 06].

Une première façon de procéder est de faire appel à des observateurs humains pour évaluer les images. Cependant, cette méthode présente plusieurs défauts : d'abord elle fait appel à la subjectivité de l'être humain, et de ce fait tous les observateurs n'évalueront pas les images de la même manière. De plus, elle ne fournit qu'un résultat qualitatif, alors que l'on aimerait avoir un résultat quantitatif.

Et enfin, l'évaluation humaine des images est fastidieuse et devient difficile à mettre en œuvre quand le nombre d'images à traiter est important. C'est pourquoi des méthodes permettant d'évaluer automatiquement la qualité d'une image compressée ont été proposées.

Toutes les méthodes développées dans ce domaine peuvent être classées en trois grandes catégories : celles qui sont à référence totale (utilisation complète de l'image originale), appelées aussi « *Full Reference* », celles qui utilisent quelques paramètres de la référence, appelées aussi « *Half Reference* » ou méthodes à référence réduite [OLS 97] et celles qui n'utilisent aucune référence [JUN 00]. Les méthodes « *Half Reference* » sont souvent utilisées dans le cadre de la transmission des images. Les méthodes « Sans Référence » se basent sur la connaissance a priori de la nature des dégradations ou/et des images.

Certaines de ces méthodes utilisent un critère mathématique comme le PSNR, le MSE ou l'entropie des histogrammes normalisés pour mesurer la distorsion entre l'image originale et l'image compressée.

Dans le domaine du codage, plusieurs études ont été faites pour analyser les mesures standards de dégradations basées sur des différences entre pixels des images originale et dégradée [ESK 95].

Dans un système de compression d'images, l'image dégradée par compression $\hat{I}$ est toujours comparée à l'originale I pour déterminer son rapport de ressemblance. A côté du critère qualitatif et subjectif, on trouve le critère quantitatif le plus utilisé qui est basé sur la mesure de l'erreur quadratique moyenne (MSE) calculée entre les pixels originaux et dégradés:

$$MSE = \sum_{n=1}^N \sum_{m=1}^M \frac{(I(m,n) - \hat{I}(m,n))^2}{M \times N}$$

Où $(M \times N) \times$ est la taille de l'image, et $I(m,n)$ et $\hat{I}(m,n)$ sont respectivement les amplitudes de pixel sur les images originale et dégradée.

IV.2.2 Méthode proposée :

a. Les paramètres influant sur la qualité de l'image reconstruite :

La précision avec laquelle l'image est reconstruite dépend de la précision de sa description LWDI. Avant de calculer la description d'une image, celle-ci doit passer tout d'abord par l'étape de segmentation et détection des contours.

La qualité de l'image reconstruite dépend des facteurs suivants :

- Forme géométriques des éléments reconstruits dans l'image: elle dépend des facteurs suivants :
 - Le contour externe d'une région décrit par le descripteur LWDI peut omettre des pixels qui ont moins d'intensité lumineuse que d'autres. Ceci est dû aux algorithmes de segmentation d'images qui malgré leurs diversité et leurs abondance, aucun algorithme ne peut fournir une bonne décomposition de l'image en régions.
 - D'une autre part la précision avec laquelle chaque région est reconstruite dépend de la précision de la description LWDOS de sa silhouette. Cette dernière est liée au nombre de points de courbure détectés tout au long du contour de la silhouette. Ce nombre dépend des deux paramètres d_{max} et α_c de l'algorithme de Chetverikov [DAH 06].
- Couleur des régions :

La description suivant le langage LWDI produit une description de la couleur moyenne des pixels d'une même région, alors que dans l'image originale ceux-ci peuvent avoir des couleurs très proches. L'amélioration dans la précision de reconstruction doit passer par la proposition d'une description qui doit tenir compte des différentes couleurs existantes dans une même région.

b. Méthode de quantification :

Quantification de la géométrie des éléments de l'image reconstruite :

Afin de quantifier la qualité des images obtenues tout en s'intéressant aux formes géométriques des régions reconstruites, nous proposons une méthode qui consiste à superposer chaque couple de silhouettes (initiale et reconstruite) et évaluer la quantité de pixels additionnels et manquants dus aux deux facteurs géométriques citées ci-dessus.

Pour ce faire, nous calculerons le taux d'erreur de reconstruction TEGR d'une région comme suit :

$$\text{TEGR} = (\text{NPA} + \text{NPM})/\text{NPR} \text{ où :}$$

- ❖ NPA (Nombre *de Pixels Additionnels*) représente le nombre de pixels en plus dans l'image reconstruite relativement au nombre de pixels dans l'image originale.
- ❖ NPM (Nombre *de Pixels Manquants*) représente le nombre de pixels en moins dans l'image reconstruite relativement au nombre de pixels dans l'image originale.
- ❖ NPR (Nombre de pixels de la région dans l'image origine)

Ces valeurs sont calculées en respectant les conditions suivantes:

- Le taux d'erreur de reconstruction TEGR est calculé pour chaque région conjointement avec l'ensemble de ses régions internes.
- Le taux d'erreur de reconstruction TEGI de l'image globale est calculé comme étant la somme des points additionnels et points manquants des différentes régions contenues dans l'image origine, divisée par le nombre total de points dans l'image.

Quantification de la qualité de la couleur de l'image reconstruite :

Le taux d'erreur dans la couleur d'une région reconstruite (TECR) est calculé en comparant les couleurs des régions originales et reconstruites:

- Pour chaque pixel commun entre la région reconstruite et d'origine, nous calculons leur différence de couleur (DC) relativement au nombre maximum de couleurs représentées par la carte graphique.
- Le taux d'erreur dans la couleur de la région TECR est la somme des 'DC' divisée par le nombre de pixels N (de la région d'origine).

Ainsi la qualité de couleur de région se mesure par $\text{QCR} = 1 - \text{TECR}$.

L'arrière plan est considéré comme étant une région et la qualité de sa couleur est calculée par le même principe que celui d'une région.

Ainsi, le taux d'erreur dans la couleur de l'image TECI reconstruite est la somme des DC de ses régions divisée par le nombre total de points dans l'image, et la qualité de couleur de l'image se mesure par $\text{QCI} = 1 - \text{TECI}$.

Quantification de la qualité globale de l'image reconstruite :

Le taux d'erreur dans la qualité (géométrie et couleur) d'une région englobe le taux d'erreur de la géométrie et celui de la couleur.

Le taux d'erreur dans la qualité de l'image TEI est la somme de celui de la géométrie et celui de la couleur. $TEI = TEGI + TECI$.

Exemple:

L'image dans la figure 3.25 est constituée de deux régions R_1 et R_2 colorées respectivement en rouge et en bleu et d'un arrière plan coloré en jaune (chaque carré représente un pixel).

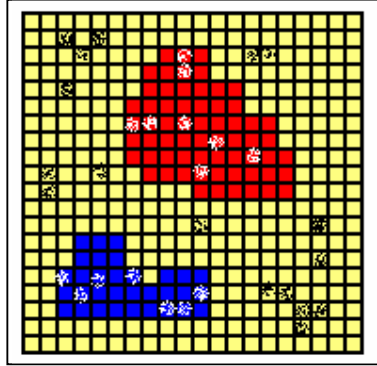


Figure 3.25 : Image d'exemple 'originale'

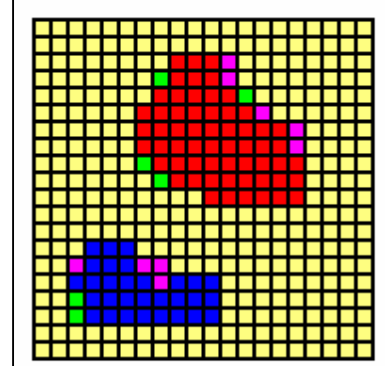


Figure 3.26 : Image d'exemple 'reconstruite'

Les carrés colorés en violet dans l'image résultat représentent les pixels en plus dans la reconstruction des régions et ceux en vert représentent les pixels en moins.

Les carrés contenant des taches sont les pixels dont leur couleur est différente de l'image initiale.

On suppose que la couleur soit codée uniquement sur 1 octet donc le nombre maximum de couleurs est 256.

Région R_1 :

Forme géométrique de la région :

Le nombre total de pixel $N_1 = 63$

Le nombre de pixel additionnels $NA_1 = 5$

Le nombre de pixel manquants $NM_1 = 3$

$TEGR_1 = (NA_1 + NM_1) / N_1 = 8 / 63 = 12,69\%$

Couleur de la région :

Soient 10, 15, 20, 12, 25, 30, 11, 9 les valeurs de DC qui correspondent aux pixels qui ont changés de couleurs.

Ainsi on $TECR_1 = (\sum DC / 256) / N_1 \cong 0,82\%$

Donc $TER_1 \cong (NA_1 + NM_1 + \sum DC / 256) / N_1 = 13,51\%$

Région R_2 :

Forme géométrique de la région :

Le nombre total de pixel $N_2 = 32$

Le nombre de pixel en plus $NA_2 = 4$

Le nombre de pixel en moins $NM_2 = 2$

$$TEGR_2 = (NA_2 + NM_2) / N_2 = 6 / 32 = 18,75\%$$

Couleur de la région :

Soit 5, 5, 10, 12, 8, 9, 11 les valeurs de DC qui correspondent aux pixels qui ont changés de couleurs.

$$\text{Ainsi on a } TEGR_2 = (\sum DC / 256) / N_2 \cong 0,73\%$$

$$\text{Donc } TER_2 \cong (NA_2 + NM_2 + \sum DC / 256) / N_2 = 19,48\%$$

L'arrière plan:

Forme géométrique:

Le nombre total de pixel $N_3 = 305$

Le nombre de pixel en plus $NP_3 = 6$

Le nombre de pixel en moins $NM_3 = 9$

$$TEGR_3 = (NA_3 + NM_3) / N_3 = 4,92\%$$

Couleur:

Soit 15, 3, 4, 20, 12, 10, 8, 9, 11, 14, 25, 30, 12, 35, 15, 10, 9 les valeurs de DC qui correspondent aux pixels qui ont changés de couleurs.

$$\text{Ainsi on a } TEGR_{arp} = (\sum DC / 256) / N_3 \cong 0,32\%$$

$$\text{Donc } TER_{arp} \cong 5,24\%$$

L'image entière:

Le nombre total de pixel $N = 400$

$$TERI = TER_1 * N_1 / N + TER_2 * N_2 / N + TER_{arp} * N_3 / N = 7.68\%$$

$$TP_{img} = 92.32\%$$

V. Conclusion:

La reconstruction de l'image à partir de sa description LWDI est l'opération inverse à celle de la description. La coopération de ces deux opérations leur permet d'être utiles dans de multiples domaines notamment dans les bases de données image, la transmission sur réseaux, et l'intégration dans le Web.

Dans ce chapitre nous avons présenté la méthode de reconstruction de l'image à partir de sa description LWDI. En effet la reconstruction de celle-ci se base sur la reconstruction de la silhouette de chaque région.

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.

EXPERIMENTATIONS

I. Introduction

Nous avons conçu une application pour la visualisation d'image qui implémente l'algorithme de reconstruction d'image développé dans le chapitre précédent en utilisant le langage DELPHI.

Cette application peut être utilisée à la fois pour la reconstruction d'une image à partir de sa description LWDI ou à la reconstruction d'une silhouette seulement (à partir d'une description LWDOS).

Dans ce chapitre nous allons expérimenter cette application sur différentes images, tout en quantifiant les résultats obtenus en utilisant la technique décrite dans le chapitre précédent.

II. Outils nécessaires:

Pour évaluer les performances de la méthode proposée nous avons effectué des tests de reconstruction de différentes images à partir de leurs descriptions. Les résultats obtenus sont ensuite évalués par une comparaison de la qualité des images reconstruites relativement aux images originales.

Pour faire la description d'une image on doit passer par les étapes suivantes:

II.1. Segmentation d'image:

La segmentation a pour objectif de découper l'image en plusieurs régions de couleurs contiguës et de faire apparaître le contour externe de chaque région. La technique choisie pour segmenter l'image est dite *JSEG*.

L'outil de segmentation qui implémente cet algorithme fournit l'image segmentée au bout de quelques instants qui varient généralement entre quelques secondes jusqu'à quelques minutes selon la complexité de l'image source.

II.2. Description de régions:

La description de l'image en langage LWDI est faite en se basant sur la description du contour externe de chaque région.

Dans notre travail nous avons utilisé un outil développé au laboratoire LRIA de l'USTHB. Cet outil permet de donner la forme général d'un descripteur LWDI, en donnant les différentes régions et régions internes (leurs couleurs et leurs positions dans l'image) et la position du descripteur LWDOS de la silhouette de chaque région (sans donner la description de celle-ci).

Donc pour avoir la description complète de l'image on doit utiliser d'autre outil pour ajouter la description des silhouettes.

II.3. Description de silhouettes:

La description LWDOS de chaque silhouette est faite en utilisant un autre outil développé dans le même institut (LRIA). Tout d'abord l'image doit être divisée en plusieurs images dont chacune contient uniquement une région. Cet outil doit être ensuite appliqué sur chaque image contenant une région pour donner la description de sa silhouette. Cette description est ensuite ajoutée au descripteur LWDI expliqué dans la section précédente.

Le descripteur obtenu après la restitution de la description de chaque silhouette constitue le descripteur LWDI complet de l'image.

Ainsi notre application se base donc sur ce descripteur afin de reconstruire l'image globale.

III. Reconstruction d'image par l'application développée:

Dans cette section nous allons utiliser l'application développée pour reconstruire quelques exemples d'images de synthèse et des images réelles, tout en évaluant les résultats obtenus (temps de reconstitution et qualité d'image) pour chaque exemple.

III.1.Présentation de l'application développée:

a. Structure de données utilisées:

En examinant la structure d'une image on peut remarquer qu'elle est structurée d'une manière hiérarchique: (l'image est formé d'un ensemble de régions, une région est formée d'une silhouette et d'un ensemble facultatif de régions internes,...), ainsi le traitement d'un élément descendant fait partie du traitement de son élément supérieur.

Donc la structure de données à utiliser doit être de préférence de la même manière pour ce bénéficier de cette hiérarchie. La meilleure structure qui reflète cette hiérarchie est sans doute celle basée sur la notion d'objet (Orienté objet).

La structure de donnée peut être vue comme étant un arbre n-aire orienté vers le bas: l'image globale constitue la racine de l'arbre, les régions situées directement dans l'image constituent les premiers nœuds, et ainsi de suite jusqu'aux contours élémentaires et segments de jonction qui constituent les feuilles de cet arbre.

Cette structure de données (orientés objets) peut être utilisée pour la vérification du descripteur et aussi pour la création de l'image.

Dans la programmation orientée objet chacun des objets dispose de certains attributs (propriétés) contenant des données simples ou structurées, et des méthodes (procédures) contenant du code permettant de son initialisation et sa mise à jour.

L'utilisation de deux structures de données différentes (une pour la vérification du descripteur et l'autre pour la reconstruction de l'image) permet d'utiliser uniquement les données nécessaires pour chaque étape.

Etape de vérification: les données nécessaires dans cette étape sont celles qui permettent de vérifier la structure de chaque objet (les balises, les noms, les différents

attributs,...). Cette structure est construite au fur et à mesure que la vérification du descripteur progresse.

Etape de reconstruction: les données nécessaires dans cette étape sont uniquement celles qui détiennent des informations nécessaires pour la reconstruction (ces données peuvent être extraites directement du descripteur ou calculées par rapport à d'autres).

b. Vérification du descripteur:

La vérification du descripteur fournit en sortie soit la première structure de données dans le cas où le descripteur est jugé juste, ou un message d'erreur qui indique l'objet contenant l'erreur.

Dans le cas où la structure de donnée ait été créée et ait été jugée correcte, l'étape de reconstruction commence tout d'abord par la vérification s'il s'agit d'une image (le descripteur commence par la balise '<DXLWDI>') ou d'une silhouette (le descripteur commence par une balise '<DXLWDOS>') et lance la procédure de reconstruction appropriée.

S'il s'agit d'une partition ou d'une partie composée (le descripteur commence par '<P1>' ou par '<CP>') un message d'erreur est affiché indiquant l'absence de l'entête, ensuite la reconstruction de l'objet commence.

c. Caractéristiques de l'application:

L'application développée peut être utilisée donc à la fois pour la reconstruction des images ou des silhouettes.

La reconstruction d'image est faite en respectant les mêmes couleurs décrites dans le descripteur LWDI, par contre en ce qui concerne les silhouettes, sachant qu'un descripteur LWDOS ne précise pas de couleur pour le fond de silhouette, ainsi ce dernier est reproduit en couleur verte.

L'application est dotée d'un compteur de temps qui permet d'évaluer le temps mis dans la reconstruction d'une image. Ce compteur permet d'évaluer le temps en milliseconde.

Vu que le temps mis dans la reconstruction des images qui ont été sujets d'expérimentation était très court (moins d'une seconde) la technique utilisée pour quantifier ce paramètre est de faire plusieurs expériences sur la même image et noter le temps mis à chaque expérience, ensuite le calcul de la moyenne de ces temps nous permet d'avoir une valeur approximative de celui-ci.

Les caractéristiques de la machine et le nombre de programmes en cours d'exécution ont certainement un effet ralentissant sur la reconstruction des images, les expériences illustrées dans ce qui suit sont exécutées dans les conditions suivantes:

- Microprocesseur: Intel 1.1 GHz.
- Capacité de la RAM: 64 MO.
- Système d'exploitation: Windows ME.
- Aucun programme n'est en cours d'exécution sauf l'application de reconstruction et les routines du système d'exploitation.

III.2. Reconstruction d'image:

Les paramètres de l'algorithme de *Chetverikov* (l'angle ' α ' et le nombre de point ' n ') ont un impact considérable sur la qualité d'image reproduite surtout pour les images contenant beaucoup de contours élémentaires courbés.

Afin de mettre en évidence cet impact nous allons utiliser pour quelques exemples d'image un descripteur crée en appliquant cet algorithme avec différents paramètres.

N.B: Généralement les valeurs utilisées pour ces deux paramètres sont $\alpha=150^\circ$ et $n=5$, et sont suffisantes pour avoir des images assez acceptable, sauf pour les images contenant des régions elliptiques.

Les images utilisées dans ces exemples comportent uniquement des régions colorées avec une seule couleur (les pixels dans chaque région ont la même couleur), ceci est dû à la limite du descripteur LWDI qui définit une couleur unique pour l'ensemble des points dans une région. Ainsi le calcul de la qualité de l'image reproduite se limite donc au calcul du taux d'erreur de la géométrie des régions TEGR (points additionnels et manquants) (la différence de couleur est toujours nulle).

a. Images de synthèse:

Afin de valider l'algorithme proposé et quantifier la qualité de reconstruction, nous avons réalisé un ensemble de tests de reconstruction de diverses images à partir de leurs descriptions LWDI et LWDOS.

1. Reconstruction de silhouette:

Les résultats obtenus dans la reconstruction de silhouette dépendent fortement du type de silhouette et de la précision du descripteur.

a. Silhouette polygonale:

Dans le cas d'une silhouette polygonale (contenant uniquement des contours élémentaires droits) l'image reproduite est généralement identique à l'originale.

L'exemple suivant illustre ce cas:

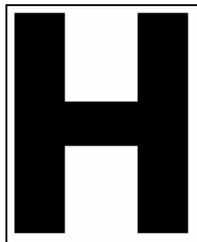


Figure 4.1 : Silhouette polygonale originale

a) L'image originale:

- Dimensions: 128 x 160.
- Nombre de points: 20480.
- Nombre de points dans la silhouette: NB=11602.

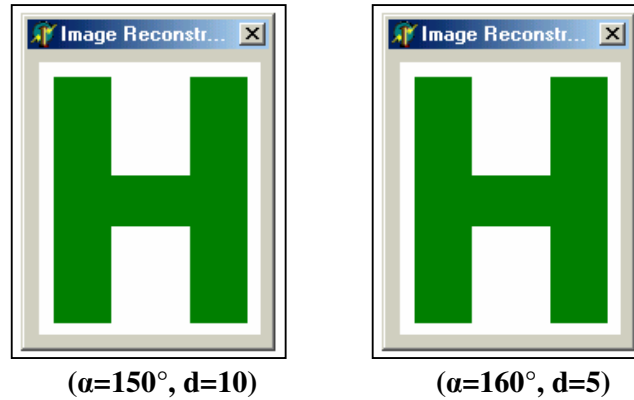


Figure 4.2: Reconstruction de silhouette1

b) Les images reconstruites:

- Nombre de contours: $N_c = 16$.
- Temps de reconstruction moyen $T_{moy1} \approx 50ms$
- Nombre de points dans la silhouette: $NB' = 11602$
- Nombre de points additionnels et manquants: $NBP^+ + NBP^- = 0$.
- Taux de perte: $(NBP^+ + NBP^-) / NB = 0\%$
- Qualité d'image = 100%

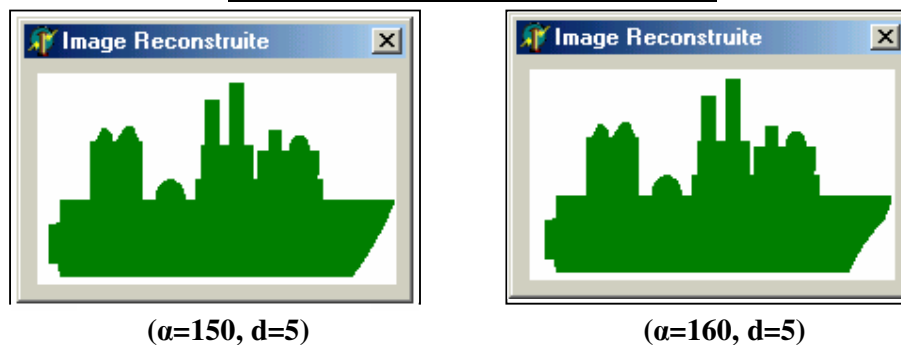
c) Comparaison:

Les deux silhouettes sont reproduites avec une qualité maximum et dans un temps minimum (aux environs de 50ms).

b. Silhouette avec quelques courbures:



Figure 4.3 : Image "Bateau" originale



a) Figure 4.4: Reconstruction de l'image "Bateau"

- Dimensions: 340 x 180.
- Nombre de points: 61200.
- Nombre de points dans la silhouette: $NB = 29993$

b) L'image reconstruite 1 ($\alpha=150$, $d=5$):

- Nombre de contours: $N_{c1}= 34$.
- Temps de reconstruction moyen $T_{moy1}\approx 50ms$
- Nombre de points dans la silhouette: $NB'=30731$
- Nombre de points additionnels et manquants: NBP^++NBP^-
 $=757+19=776$.
- Taux de perte: $TP_1=(NBP^++NBP^-)/NB=2,59\%$
- Qualité d'image = $97,41\%$

c) L'image reconstruite 2 ($\alpha=160$, $d=5$):

- Nombre de contours: $N_{c2}= 46$.
- Temps de reconstruction moyen $T_{moy2}\approx 55ms$
- Nombre de points dans la silhouette: $NB''=30718$
- Nombre de points additionnels et manquants: NBP^++NBP^-
 $=738+13=751$.
- Taux de perte: $TP_2=(NBP^++NBP^-)/NB=2,50\%$
- Qualité d'image = $97,50\%$

d) Comparaison:

Le temps de reconstruction des deux images est pratiquement le même et reste aux environs de 50ms. En ce qui concerne la qualité d'image, les deux images sont reproduites avec une qualité meilleure (presque la même pour les deux images).

En effet le changement des deux paramètres α et d n'a pas donné beaucoup d'influence sur la qualité ou sur le temps de reconstruction. Ceci est dû essentiellement à la nature des silhouettes (contiennent peu de contours courbés).

2. Reconstruction d'images:

Nous allons étudier la reconstruction de deux images dessinées à la main en utilisant un logiciel du traitement d'image.

La première image est constituée de quelques régions simples, tandis que la deuxième est plus complexe que ce soit dans le nombre de régions dans l'image ou dans leurs structures.

a. Image contenant des régions simples:

1) Description de l'image originale:

- Dimensions: 210×200 .
- Nombre de points: $NB=42000$ points.
- Nombre de régions 3.

a. Région $R_{0,1}$ (banane):

Nombre de points: $N_1=5838$ points

b. Région $R_{0,2}$ (poire):

Nombre de points $N_2=4647$ points

c. Région $R_{0,3}$ (pomme):

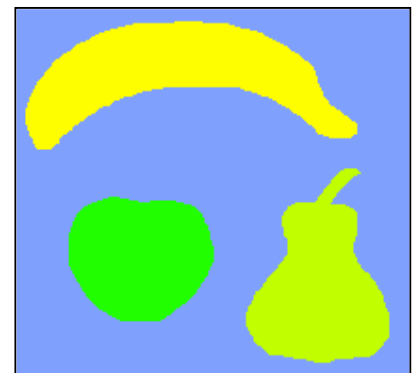


Figure 4.5: Image "Fruits" originale

Nombre de points $N_3 = 4069$ points

d. Arrière plan:

Nombre de points $N_0 = 27446$ points

2) Images reconstruites:

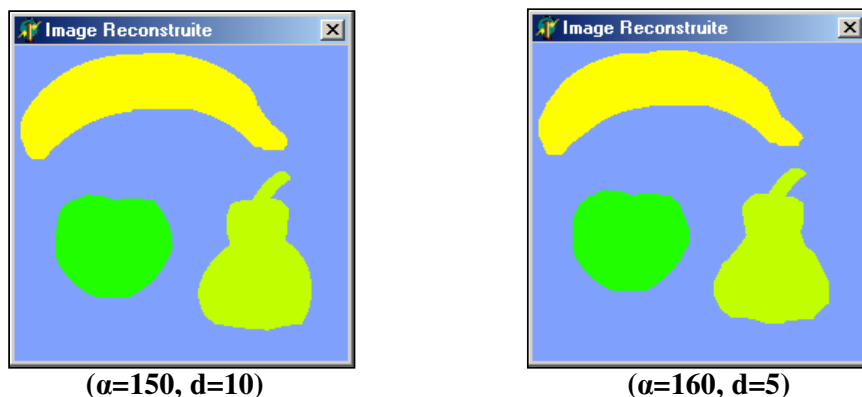


Figure 4.6: Reconstruction de l'image « fruit »

a) Image 1 (d=10, α=150):

– Nombre de contour $N_{c1} = 37$ contours.

1. Temps de reconstruction:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	60	50	60	50	50	50	50	60	50	50	60	50	50	60	60

Le temps moyen de reconstruction est $T_{moy1} \approx 55$ ms.

2. Qualité d'image:

a. Région1 "Banane":

• Nombre de points:

– NB'_1 : 6227 points.

• Nombre de points additionnels et manquants:

– $N'_1 = NBP^+ + NBP^- = 399 + 10 = 409$.

• Taux de perte ($TEGR'_1$):

– $TEGR'_1 = N'_1 / NB_1 * 100 = 409 / 5838 * 100 = 7,05\%$

b. Région2 "Poire":

• Nombre de points:

– NB'_2 : 4945 points.

• Nombre de points additionnels et manquants:

– $N'_2 = NBP^+ + NBP^- = 377 + 79 = 456$.

- Taux de perte (TEGR'₂):
 - $TEGR'_2 = N'_2 / NB_2 * 100 = 456 / 4647 * 100 = 9,81\%$

c. Région3 "Pomme":

- Nombre de points:
 - NB'_3 : 4241 points.
- Nombre de points additionnels et manquants:
 - $N'_3 = NBP^+ + NBP^- = 203 + 31 = 234$.
- Taux de perte (TEGR'₃):
 - $TEGR'_3 = N'_3 / NB_3 * 100 = 234 / 4069 * 100 = 5,75\%$

d. Arrière plan:

- Nombre de points:
 - NB'_0 : nombre de point dans l'image reconstruite = 26587 points.
- Nombre de points additionnels et manquants:
 - $N'_0 = NBP^+ + NBP^- = 120 + 979 = 1099$.
- Taux de perte (TEGR'₀):
 - $TEGR'_0 = DC / N'_0 * 100 = 1099 / 27446 * 100 = 4\%$.

e. Image complète:

- Nombre de points colorés:
 - $NBC' = NB + NB'_1 + NB'_2 + NB'_3 = 57413$ points.
- Taux de perte:
 - $TERI = (N'_1 + N'_2 + N'_3) / NB = 5,23\%$

Donc l'image est reproduite avec une qualité de **94,77%**.

b) Image 2 (d=5, α=160):

- Nombre de contour $N_{c2} = 57$.

1. Temps de reconstruction:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	60	50	50	50	60	60	50	50	60	50	50	60	50	60	50

Le temps moyen de reconstruction $T_{moy2} \approx 55$ ms

2. Qualité d'image :

a. Région1 "Banane":

- Nombre de points:
 - NB''_1 : 6179 points.
- Nombre de points additionnels et manquants:

- $N''_1 = NBP^+ + NBP^- = 359 + 18 = 377$.
- Taux de perte (TEGR''₁):
 - $TEGR''_1 = N''_1 / NB_1 * 100 = 377 / 5838 * 100 = 6,46\%$

b. Région2 "Poire":

- Nombre de points:
 - NB''_2 : 4903 points.
- Nombre de points additionnels et manquants:
 - $N''_2 = NBP^+ + NBP^- = 279 + 23 = 302$.
- Taux de perte (TEGR''₂):
 - $TEGR''_2 = N''_2 / NB_2 * 100 = 302 / 4647 * 100 = 6,50\%$

c. Région3 "Pomme":

- Nombre de points:
 - NB''_3 : 4253 points.
- Nombre de points additionnels et manquants:
 - $N''_3 = NBP^+ + NBP^- = 202 + 18 = 220$.
- Taux de perte (TEGR''₃):
 - $TEGR''_3 = N''_3 / NB_3 * 100 = 220 / 4069 * 100 = 5,41\%$

d. Arrière plan:

- Nombre de points:
 - NB''_0 : nombre de point dans l'image reconstruite = 26665 points.
- Nombre de points additionnels et manquants:
 - $N''_0 = NBP^+ + NBP^- = 59 + 840 = 899$.
- Taux de perte (TEGR''₀):
 - $TEGR''_0 = N''_0 / NB_0 * 100 = 899 / 27446 * 100 = 3,27\%$.

e. Image entière:

- Nombre de points colorés:
 - $NBC'' = NB + NB''_1 + NB''_2 + NB''_3 = 57335$ points.
- Taux de perte:
 - $TERI'' = (N''_1 + N''_2 + N''_3) / NB = 4,28\%$

Donc l'image est reproduite avec une qualité de **95,72%**.

3) Comparaison:

Image reconstruite	Nombre de contours	Nombre de points colorés	Temps moyen de reconstruction (ms)	Qualité
Image(1) ($\alpha=150$, $d=10$)	37	57413	55	94,77%
Image(2) ($\alpha=160$, $d=5$)	57	57335	55	95,72%

En comparant les résultats obtenus sur les deux images on remarque que le temps de reconstruction est le même pour les deux images, tandis que la qualité de la 2^{ème} image a reconnu une petite hausse. En fait cette similarité est due à la simplicité de l'image et à la différence minimale dans le nombre de contours et dans le nombre de points colorés dans les deux images.

b. Image avec des régions complexes:

1) Description de l'image originale:

- Longueur: 300 pixels.
- Hauteur: 345 pixels
- Nombre total de points NB=103500.
- Nombre de régions et régions internes: 12.

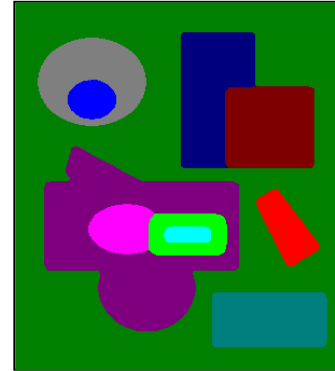
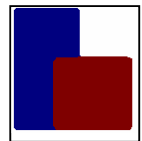


Figure 4.7: Image complexe

a. La région $R_{1,0}$:

• Nombre de points :

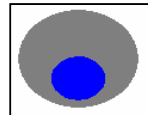
- $NB_1 = 12854$ point dont 6600 points en bleu ($R_{0,0}$) et 6254 points en marron ($R_{0,1}$)



b. La région $R_{1,1}$:

• Nombre de points :

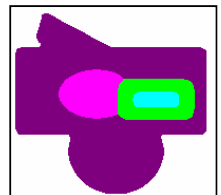
- $NB_2 = 6454$ point dont 5161 points en gris et 1293 points en bleu ($R_{0,2}$)



c. La région $R_{3,0}$:

• Nombre de points :

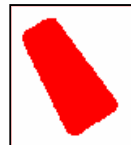
- $NB_3 = 20468$ point dont 15506 points en violet, 2212 points en rose ($R_{0,3}$), 2123 points en vert et 627 en bleu ciel ($R_{0,4}$).



d. La région $R_{0,5}$:

• Nombre de points :

- $NB_4 = 2121$ points.



e. La région $R_{0,6}$:

• Nombre de points :

- $NB_5 = 5432$ points.



f. Arrière plan:

- $NB_0 = 56171$ points.

2) Images reconstruites:

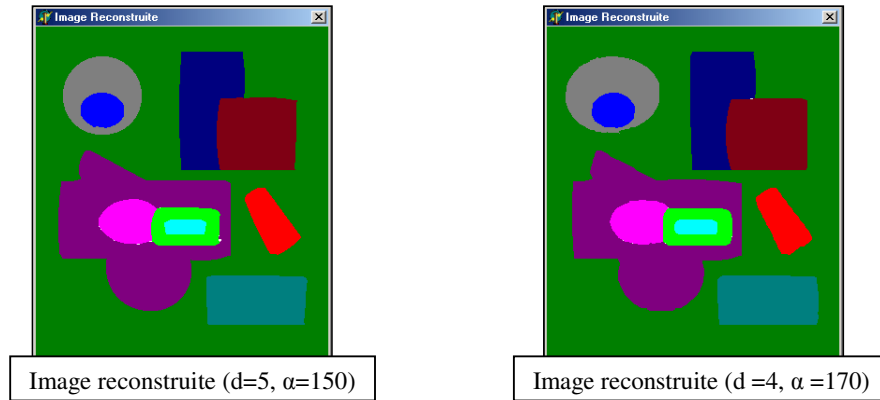


Figure 4.8: Reconstruction d'image complexe

a) Image 1 (d=5, α=150):

- Nombre de contours $N_{c1} = 89$.

1. Temps de reconstruction:

Le tableau suivant donne le temps mis dans la reconstruction de l'image:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	60	50	60	110	50	60	60	100	100	50	50	60	100	50	60

Le temps moyen de reconstruction $T_{moy1} \approx 70$ ms.

2. Qualité d'image:

a. Région $R_{1,0}$:

• Nombre de points dans les régions:

- NB'_1 : 12977 points dont 6537 points en bleu et 6440 points marron.

• Nombre de points additionnels et manquants:

- Région bleue (N'_b): $NBP^+ + NBP^- = 129 + 192 = 321$
- Région marron (N'_m): $NBP^+ + NBP^- = 223 + 37 = 260$

• Taux de perte ($TERG'_1$):

- $TERG'_1 = (N'_b + N'_m) / NB_1 * 100 = 263 / 12878 * 100 = 4,52\%$

Donc le taux de précision est 95,48%

b. Région $R_{1,1}$:

• Nombre de points:

- NB'_2 : 5616 points dont 4227 points en gris et 1389 en bleu.

• Nombre de points additionnels et manquants:

- Région grise (N'_g): $NBP^+ + NBP^- = 74 + 1008 = 1182$
- Région bleue (N'_b): $NBP^+ + NBP^- = 103 + 7 = 110$

• Taux de perte ($TERG'_2$):

– $TERG'_2 = (N'_g + N'_b) / NB_2 * 100 = 1292 / 6454 * 100 = 20,02\%$.
Donc le taux de précision est 79,98%

c. Région R_{3,0}:

• Nombre de points:

– NB'₃: 20952 points dont 15787 en violet et 2298 en rose, 2190 en vert et 677 en bleu ciel).

• Nombre de points additionnels et manquants:

– Région violet (N'_t): $NBP^+ + NBP^- = 601 + 338 = 939$.
– Région rose (N'_r): $NBP^+ + NBP^- = 118 + 32 = 150$.
– Région verte (N'_v): $NBP^+ + NBP^- = 152 + 85 = 237$.
– Région bleu ciel (N'_b): $NBP^+ + NBP^- = 59 + 9 = 68$.

• Taux de perte (TP'₃):

– $TP'_3 = (N'_t + N'_r + N'_v + N'_b) / NB_3 * 100 = 1394 / 20468 * 100 = 06,81\%$
Donc le taux de précision est 93,19%.

d. Région R_{0,3}:

• Nombre de points:

– NB'₄: =2298 points.

• Nombre de points additionnels et manquants:

– N'₄: $NBP^+ + NBP^- = 189 + 12 = 201$.

• Taux de perte (TERG'₄):

– $TERG'_4 = N'_4 / NB_4 * 100 = 201 / 2121 * 100 = 09,48\%$.
Donc le taux de précision est 90,52%.

e. Région R_{0,6}:

• Nombre de points:

– NB'₅: =5556 points.

• Nombre de points additionnels et manquants:

– N'₅: $NBP^+ + NBP^- = 219 + 95 = 314$.

• Taux de perte (TERG'₅):

– $TERG'_5 = N'_5 / NB_5 * 100 = 314 / 5432 * 100 = 5,78\%$.
Donc le taux de précision est 94,22%.

f. Arrière plan:

• Nombre de points:

– NB'₀: =56101 points.

• Nombre de points additionnels et manquants:

– N'₀: $NBP^+ + NBP^- = 1173 + 1243 = 2416$.

• Taux de perte (TERG'₀):

– $TERG'_0 = N'_0 / NB_0 * 100 = 2416 / 56171 = 4,30\%$.
Donc le taux de précision est 95,70%.

g. L'image globale:

- Nombre de points colorés:

- NB= 171107 points.

- Taux de perte global (TERI):

- $TERI = (N'_0 + N'_1 + N'_2 + N_3^* + N'_4 + N'_5 + N'_6) / NB = 5.99\%$

Donc l'image est reproduite avec une précision de **94,01 %**.

b) Image 2 (d=4, $\alpha=170$):

- Nombre de contours Nc= **258**.

1. Temps de reconstruction:

Les temps obtenus dans les différentes expériences sont indiqués dans le tableau suivant:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	110	100	110	50	100	60	110	50	60	100	60	110	50	60	100

Le temps moyen de reconstruction $T_{moy2} \approx 75$ ms.

2. Qualité d'image:

a. Région $R_{1,0}$:

- Nombre de points dans les régions:

- NB''_1 : 13015 points dont 6489 points en bleu et 6523 points marron.

- Nombre de points additionnels et manquants:

- Région bleue (N''_b): $NBP^+ + NBP^- = 146 + 257 = 403$

- Région marron (N''_m): $NBP^+ + NBP^- = 269 + 0 = 269$

- Taux de perte ($TERG''_1$):

- $TERG''_1 = (N''_b + N''_m) / NB_1 * 100 = 672 / 12878 * 100 = 5,22\%$

Donc le taux de précision est 94,88%

b. Région $R_{1,1}$:

- Nombre de points:

- NB''_2 : 6519 points dont 5134 points en gris et 1385 en bleu.

- Nombre de points additionnels et manquants:

- Région grise (N''_g): $NBP^+ + NBP^- = 240 + 267 = 507$

- Région bleue (N''_b): $NBP^+ + NBP^- = 98 + 6 = 104$

- Taux de perte ($TERG''_2$):

- $TERG''_2 = (N''_g + N''_b) / NB_2 * 100 = 611 / 6454 * 100 = 9,47\%$.

Donc le taux de précision est 90,53%

c. Région $R_{3,0}$:

- Nombre de points:

- NB''_3 : 20788 points dont 15587 en violet et 2326 en rose, 2170 en vert et 705 en bleu ciel.
- Nombre de points additionnels et manquants:
 - Région violet (N''_t): $NBP^+ + NBP^- = 383 + 307 = 690$.
 - Région rose (N''_r): $NBP^+ + NBP^- = 124 + 10 = 125$.
 - Région verte (N''_v): $NBP^+ + NBP^- = 125 + 78 = 203$.
 - Région bleu ciel (N''_b): $NBP^+ + NBP^- = 78 + 0 = 78$.
- Taux de perte ($TERG''_3$):
 - $TERG''_3 = (N''_t + N''_r + N''_v + N''_b) / NB_3 * 100 = 1096 / 20468 * 100 = 05,35\%$

Donc le taux de précision est 94,75%.

d. Région $R_{0,3}$:

- Nombre de points:
 - NB''_4 : =2281 points.
- Nombre de points additionnels et manquants:
 - N''_4 : $NBP^+ + NBP^- = 165 + 5 = 170$.
- Taux de perte ($TERG''_4$):
 - $TERG''_4 = N''_4 / NB_4 * 100 = 170 / 2121 * 100 = 08,02\%$.

Donc le taux de précision est 91,98%.

e. Région $R_{0,6}$:

- Nombre de points:
 - NB''_5 : =5617points.
- Nombre de points additionnels et manquants:
 - N''_5 : $NBP^+ + NBP^- = 207 + 22 = 229$.
- Taux de perte ($TERG''_5$):
 - $TERG''_5 = N''_5 / NB_5 * 100 = 229 / 5432 * 100 = 4,22\%$.

Donc le taux de précision est 95,88%.

f. Arrière plan:

- Nombre de points:
 - NB''_0 : =54979 points.
- Nombre de points additionnels et manquants:
 - N''_6 : $NBP^+ + NBP^- = 298 + 1190 = 1488$.
- Taux de perte ($TERG''_0$):
 - $TERG''_0 = N''_6 / NB_0 * 100 = 1488 / 56171 = 2,65\%$.

Donc le taux de précision est 95,70%.

g. L'image globale:

- Nombre de points colorés:
 - $NB = 172023$ points.
- Taux de perte global:

$$- \quad \text{TERI} = (N''_0 + N''_1 + N''_2 + N''_3 + N''_4 + N''_5 + N''_6) / \text{NB} = 4.12\%.$$

Donc l'image est reproduite avec une précision de **95,88%**.

3) Comparaison:

Image reconstruite	Nombre de contours	Nombre de points colorés	Temps moyen de reconstruction (ms)	Qualité
Image(1) ($\alpha=150$, $d=10$)	89	171107	65	94,01%
Image(2) ($\alpha=170$, $d=4$)	258	172023	75	95,88%

En comparant les résultats obtenus pour les deux images reconstruites, on remarque:

Le temps de reconstruction pour la première image est aux environs de 65 ms, par contre que dans la deuxième il passe aux environs de 75 ms, ceci est dû à l'augmentation du nombre de contours dans la deuxième.

La qualité d'image reconstruite est passée de 94,01% dans la 1^{ère} image à 95,88% dans la 2^{ème}, ceci est dû aussi à l'augmentation du nombre de contours.

b. Images réelles:

1. Exemple1: Logo 'A&E':

1) Description de l'image originale:

- Longueur: 462 pixels.
- Hauteur: 337 pixels
- Nombre total de points NB=155694.
- Nombre de régions et régions internes: 7.

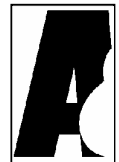


Figure 4.9: LOGO "A&E" original

a. La région $R_{1,0}$:

• Nombre de points :

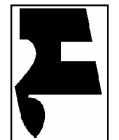
- $\text{NB}_1 = 35053$ points dont 34276 points en noir et 777 points en blanc.



b. La région $R_{0,0}$:

• Nombre de points :

- $\text{NB}_2 = 24823$ points.



c. La région $R_{0,1}$:

• Nombre de points :

- $\text{NB}_3 = 1723$ points.



d. La région $R_{0,3}$:

- Nombre de points :
 - $NB_4 = 2702$ points.



e. La région $R_{0,4}$:

- Nombre de points :
 - $NB_5 = 9087$ points.



f. La région $R_{0,5}$:

- Nombre de points :
 - $NB_6 = 193$ points.



g. Arrière plan:

- Nombre de points:
 - $NB_0 = 82113$ points.

2) Images reconstruites:

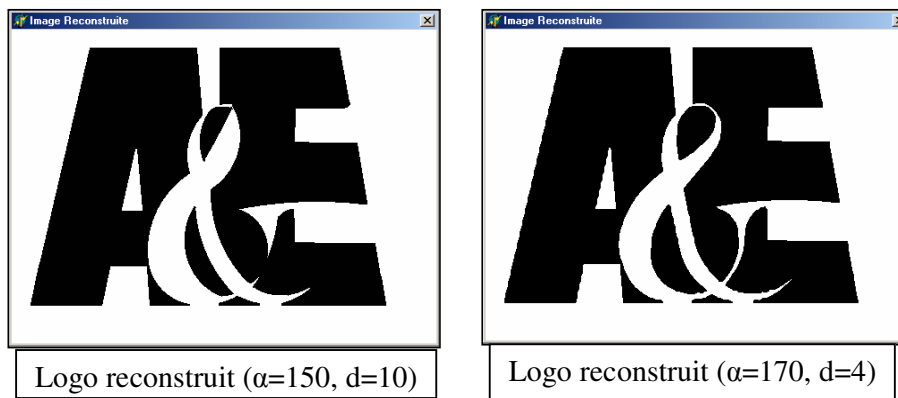


Figure 4.10: LOGO "A&E" reconstitue

a) Image 1 ($\alpha=150$, $d=10$):

- Nombre de contours $N_{c1} = 50$.

1. Temps de reconstruction:

Les temps obtenus dans les différentes expériences sont indiqués dans le tableau suivant:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	110	110	110	110	100	160	110	110	110	100	110	110	110	110	110

Le temps moyen de reconstruction $T_{moy1} \approx 112$ ms

2. Qualité d'image:

a. Région $R_{1,0}$:

- Nombre de points dans la région:
 - NB'_1 : 35699 points dont 34826 points en noir et 873 points blanc.
- Nombre de points additionnels et manquants:
 - Région Noire (N'_n): $NBP^+ + NBP^- = 697 + 147 = 844$.
 - Région Blanche (N'_c): $NBP^+ + NBP^- = 111 + 15 = 126$.
- Taux de perte ($TERG'_1$):
 - $TERG'_1 = (N'_n + N'_c) / NB_1 * 100 = 970 / 35053 * 100 = 2,77\%$

Donc le taux de précision est 97,23%

b. Région $R_{0,0}$:

- Nombre de points dans la région:
 - NB'_2 : 25869 points.
- Nombre de points additionnels et manquants:
 - N'_2 : $NBP^+ + NBP^- = 1405 + 359 = 403$
- Taux de perte ($TERG'_2$):
 - $TERG'_2 = N'_2 / NB_2 * 100 = 1794 / 24823 * 100 = 7,11\%$

Donc le taux de précision est 94,88%

c. Région $R_{0,1}$:

- Nombre de points dans la région:
 - NB'_3 : 1093 points.
- Nombre de points additionnels et manquants:
 - N'_3 : $NBP^+ + NBP^- = 311 + 941 = 1252$
- Taux de perte ($TERG'_3$):
 - $TERG'_3 = N'_3 / NB_3 * 100 = 1252 / 1723 * 100 = 72,66\%$

Donc le taux de précision est 27,34%

d. Région $R_{0,3}$:

- Nombre de points dans la région:
 - NB'_4 : 1923 points.
- Nombre de points additionnels et manquants:
 - N'_4 : $NBP^+ + NBP^- = 80 + 859 = 939$.
- Taux de perte ($TERG'_4$):
 - $TERG'_4 = N'_4 / NB_4 * 100 = 939 / 2702 * 100 = 34,75\%$

Donc le taux de précision est 65,25%

e. Région $R_{0,4}$:

- Nombre de points dans la région:
 - NB'_5 : 9546 points.
- Nombre de points additionnels et manquants:
 - N'_5 : $NBP^+ + NBP^- = 531 + 72 = 603$.

- Taux de perte (TERG'₅):

– $TERG'_5 = N'_5/NB_5 * 100 = 603/9087 * 100 = 6,64\%$

Donc le taux de précision est 93,36%

f. Région R_{0,5}:

- Nombre de points dans la région:

– $NB'_6: 299$ points.

- Nombre de points additionnels et manquants:

– $N'_6: NBP^+ + NBP^- = 106 + 0 = 106$.

- Taux de perte (TERG'₆):

– $TERG'_6 = N'_6/NB_6 * 100 = 106/193 * 100 = 54,92\%$

Donc le taux de précision est 45,07%

g. Arrière plan:

- Nombre de points dans la région:

– $NB'_0: 81277$ points.

- Nombre de points additionnels et manquants:

– $N'_0: NBP^+ + NBP^- = 2167 + 3003 = 5170$

- Taux de perte (TERG'₀):

– $TERG'_0 = N'_0/NB_0 * 100 = 5170/82113 * 100 = 6,30\%$

Donc le taux de précision est 93,70%

h. L'image globale:

- Nombre de points colorés:

– $NBC' = 230123$.

- Taux de perte global:

– $TERI = (N'_0 + N'_1 + N'_2 + N'_3 + N'_4 + N'_5 + N'_6) / NB = 6,94\%$

Donc l'image est reproduite avec une précision de **93,06%**.

b) Image 2 (α=170, d=4):

– Nombre de contours $N_c = 355$.

1. Temps de reconstruction:

Les temps obtenus dans les différentes expériences sont indiqués dans le tableau suivant:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	160	110	110	110	100	160	110	110	110	110	160	110	110	110	110

Le temps moyen de reconstruction est $T_{moy2} \approx 119$ ms

2. Qualité d'image:

a. Région R_{1,0}:

- Nombre de points dans la région:
 - NB''_1 : 35690 points dont 34831 points en noir et 858 points blanc.
- Nombre de points additionnels et manquants:
 - Région Noire (N''_n): $NBP^+ + NBP^- = 653 + 98 = 751$
 - Région Blanche (N''_c): $NBP^+ + NBP^- = 89 + 6 = 95$.
- Taux de perte ($TERG''_1$):
 - $TERG''_1 = (N''_n + N''_c) / NB_1 * 100 = 846 / 35053 * 100 = 2,41\%$

Donc le taux de précision est 97,59%

b. Région $R_{0,0}$:

- Nombre de points dans la région:
 - NB''_2 : 25468 points.
- Nombre de points additionnels et manquants:
 - N''_2 : $NBP^+ + NBP^- = 867 + 222 = 889$
- Taux de perte ($TERG''_2$):
 - $TERG''_2 = N''_2 / NB_2 * 100 = 889 / 24823 * 100 = 4,39\%$

Donc le taux de précision est 95,61%

c. Région $R_{0,1}$:

- Nombre de points dans la région:
 - NB'_3 : 1861 points..
- Nombre de points additionnels et manquants:
 - N''_3 : $NBP^+ + NBP^- = 140 + 2 = 142$
- Taux de perte ($TERG''_3$):
 - $TERG''_3 = N''_3 / NB_3 * 100 = 142 / 1723 * 100 = 8,24\%$

Donc le taux de précision est 91,76%

d. Région $R_{0,3}$:

- Nombre de points dans la région:
 - NB''_4 : 2882 points.
- Nombre de points additionnels et manquants:
 - N''_4 : $NBP^+ + NBP^- = 186 + 6 = 292$
- Taux de perte ($TERG''_4$):
 - $TERG''_4 = N''_4 / NB_4 * 100 = 292 / 2702 * 100 = 7,11\%$

Donc le taux de précision est 92,89%

e. Région $R_{0,4}$:

- Nombre de points dans la région:
 - NB''_5 : 9543 points.
- Nombre de points additionnels et manquants:
 - N''_5 : $NBP^+ + NBP^- = 463 + 7 = 470$
- Taux de perte ($TERG''_5$):

– $TERG''_5 = N''_5 / NB_5 * 100 = 470 / 9087 * 100 = 5,17\%$
Donc le taux de précision est 94,83%

f. Région $R_{0,5}$:

• Nombre de points dans la région:

– NB''_6 : 264 points.

• Nombre de points additionnels et manquants:

– N''_6 : $NBP^+ + NBP^- = 71 + 0 = 71$

• Taux de perte ($TERG''_6$):

– $TERG''_6 = N''_6 / NB_6 * 100 = 71 / 193 * 100 = 36,79\%$

Donc le taux de précision est 63,21%

g. Arrière plan:

• Nombre de points dans la région:

– NB''_0 : 79985 points.

• Nombre de points additionnels et manquants:

– N''_0 : $NBP^+ + NBP^- = 247 + 2375 = 2622$

• Taux de perte ($TERG''_0$):

– $TERG''_0 = N''_0 / NB_0 * 100 = 2622 / 82113 * 100 = 3,20\%$

Donc le taux de précision est 96,80%

i. L'image globale:

• Nombre de points colorés:

– $NBC'' = 231402$.

• Taux de perte global:

– $TERI = (N''_0 + N''_1 + N''_2 + N''_3 + N''_4 + N''_5 + N''_6) / NB = 3,49\%$

Donc l'image est reproduite avec une précision de **96,51%**.

3) Comparaison:

Image reconstruite	Nombre de contours	Nombre de points colorés	Temps moyen de reconstruction (ms)	Qualité
Image(1) ($\alpha=150$, $d=10$)	50	230123	112	93,06%
Image(2) ($\alpha=170$, $d=4$)	355	231402	119	96,51%

Le temps de reconstruction pour la 1^{ère} image dont le nombre de contours (50) est plus petit que dans la deuxième (355) est plus petit que celui de cette dernière, mais vu la rapidité des calculs celle-ci est loin d'être considérable.

La qualité d'image est aussi augmenté dans la deuxième image (96,51%) par rapport à la première (93,06%). Ceci est lié au nombre de contours élevé dans la deuxième que dans la première.

2. Exemple2: Logo '7UP':

1) Description de l'image originale:

- Longueur: 942 pixels.
- Hauteur: 625 pixels
- Nombre total de points NB=588750.
- Nombre de régions et régions internes: 11.



Figure 4.11: LOGO "7UP" original

a. La région 1:

• Nombre de points :

- $NB_1 = 77296$ points.



b. La région 2:

• Nombre de points :

- $NB_2 = 51820$ points dont 46939 points en vert, 2860 points en blanc, et 2021 points en bleu.



c. La région 3:

• Nombre de points :

- $NB_3 = 1049$ points.



d. La région 4:

• Nombre de points :

- $NB_4 = 11591$ points.



e. La région 5:

• Nombre de points :

- $NB_5 = 54994$ points.



f. La région 6:

• Nombre de points :

- $NB_6 = 12214$ points.



g. La région 7:



- Nombre de points :
 - $NB_7 = 3324$ points.

h. La région 8:

- Nombre de points :
 - $NB_8 = 2003$ points.



i. La région 9:

- Nombre de points :
 - $NB_9 = 4451$ points.



j. La région 10:

- Nombre de points :
 - $NB_{10} = 34065$ points.



k. La région 11:

- Nombre de points :
 - $NB_{11} = 6369$ points.



l. Arrière plan:

- Nombre de points:
 - $NB_0 = 328945$ points.

2) Images reconstruites:

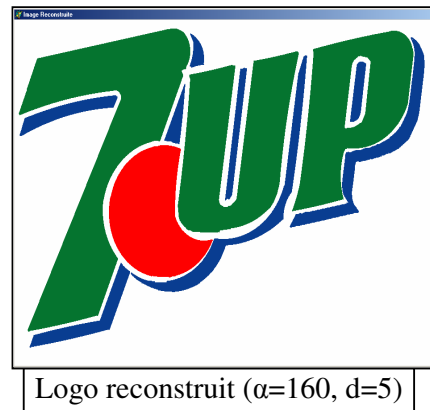
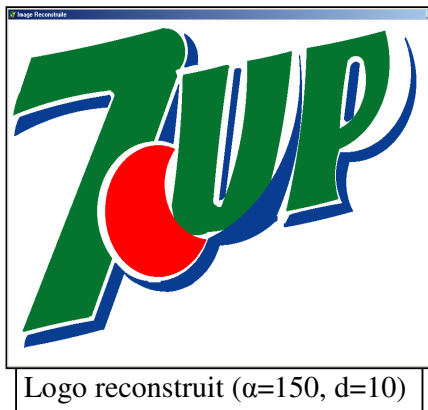


Figure 4.12: LOGO "7UP" reconstruit

a) Image 1 ($\alpha=150, d=10$):

- Nombre de contours $N_{c1} = 88$.

1. Temps de reconstruction:

Les temps obtenus dans les différentes expériences sont indiqués dans le tableau suivant:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	170	220	170	220	160	220	160	160	170	160	220	160	160	160	170

Le temps moyen de reconstruction est $T_{\text{moyl}} \approx 180 \text{ ms}$

2. Qualité d'image:

a. Région 1:

- Nombre de points dans la région:
 - NB'_1 : 78614 points.
- Nombre de points additionnels et manquants:
 - N'_1 : $NBP^+ + NBP^- = 1790 + 472 = 2262$.
- Taux de perte ($TERG'_1$):
 - $TERG'_1 = N'_1 / NB_1 * 100 = 2262 / 77296 * 100 = 2,93\%$

Donc le taux de précision est 97,07%

b. Région 2:

- Nombre de points dans la région:
 - NB'_2 : 44178 points dont 39057 en vert, 2880 points en blanc et 2241 points en bleu.
- Nombre de points additionnels et manquants:
 - Région verte (N'_v): $NBP^+ + NBP^- = 1136 + 9018 = 10154$
 - Région blanche (N'_c): $NBP^+ + NBP^- = 585 + 565 = 1150$
 - Région bleue (N'_b): $NBP^+ + NBP^- = 229 + 9 = 238$
- Taux de perte ($TERG'_2$):
 - $TERG'_2 = (N'_v + N'_c + N'_b) / NB_2 * 100 = 11542 / 51820 * 100 = 22,27\%$

Donc le taux de précision est 77,73%

c. Région 3:

- Nombre de points dans la région:
 - NB'_3 : 1176 points.
- Nombre de points additionnels et manquants:
 - N'_3 : $NBP^+ + NBP^- = 193 + 66 = 259$
- Taux de perte ($TERG'_3$):
 - $TERG'_3 = N'_3 / NB_3 * 100 = 259 / 1049 * 100 = 24,69\%$

Donc le taux de précision est 75,31%

d. Région 4:

- Nombre de points dans la région:
 - NB'_4 : 10345 points.

• Nombre de points additionnels et manquants:

– $N'_4: NBP^+ + NBP^- = 1256 + 2503 = 3759.$

• Taux de perte (TERG'₄):

– $TERG'_4 = N'_4 / NB_4 * 100 = 3759 / 11591 * 100 = 32,43\%$

Donc le taux de précision est 67,57%

e. Région 5:

• Nombre de points dans la région:

– $NB'_5: 54487$ points.

• Nombre de points additionnels et manquants:

– $N'_5: NBP^+ + NBP^- = 2810 + 3317 = 6127.$

• Taux de perte (TERG'₅):

– $TERG'_5 = N'_5 / NB_5 * 100 = 6127 / 54994 * 100 = 11,14\%$

Donc le taux de précision est 88,86%

f. Région 6:

• Nombre de points dans la région:

– $NB'_6: 17003$ points.

• Nombre de points additionnels et manquants:

– $N'_6: NBP^+ + NBP^- = 7508 + 2719 = 10227.$

• Taux de perte (TERG'₆):

– $TERG'_6 = N'_6 / NB_6 * 100 = 10227 / 12214 * 100 = 83,73\%$

Donc le taux de précision est 16,27%

g. Région 7:

• Nombre de points dans la région:

– $NB'_7: 3487$ points.

• Nombre de points additionnels et manquants:

– $N'_7: NBP^+ + NBP^- = 226 + 63 = 289.$

• Taux de perte (TERG'₆):

– $TERG'_7 = N'_7 / NB_7 * 100 = 289 / 3324 * 100 = 8,69\%$

Donc le taux de précision est 91,31%

h. Région 8:

• Nombre de points dans la région:

– $NB'_8: 2100$ points.

• Nombre de points additionnels et manquants:

– $N'_8: NBP^+ + NBP^- = 176 + 79 = 255.$

• Taux de perte (TERG'₈):

– $TERG'_8 = N'_8 / NB_8 * 100 = 255 / 2003 * 100 = 12,73\%$

Donc le taux de précision est 87,27%

i. Région 9:

- Nombre de points dans la région:

- NB'₉: 4885 points.

- Nombre de points additionnels et manquants:

- N'₉: $NBP^+ + NBP^- = 784 + 350 = 1134$.

- Taux de perte (TERG'₉):

- $TERG'_9 = N'_9 / NB_9 * 100 = 1134 / 4451 * 100 = 25,47\%$

Donc le taux de précision est 74,53%

j. Région 10:

- Nombre de points dans la région:

- NB'₁₀: 33399 points.

- Nombre de points additionnels et manquants:

- N'₁₀: $NBP^+ + NBP^- = 1713 + 2379 = 4092$.

- Taux de perte (TERG'₁₀):

- $TERG'_{10} = N'_{10} / NB_{10} * 100 = 4092 / 34065 * 100 = 12,01\%$

Donc le taux de précision est 87,66%

k. Région 11:

- Nombre de points dans la région:

- NB'₁₁: 7037 points.

- Nombre de points additionnels et manquants:

- N'₁₁: $NBP^+ + NBP^- = 727 + 59 = 786$.

- Taux de perte (TERG'₁₁):

- $TERG'_{11} = N'_{11} / NB_{11} * 100 = 786 / 6369 * 100 = 12,34\%$

Donc le taux de précision est 87,66%

l. Arrière plan:

- Nombre de points dans la région:

- NB'₀: 332033 points.

- Nombre de points additionnels et manquants:

- N'₀: $NBP^+ + NBP^- = 18710 + 15622 = 34332$

- Taux de perte (TERG'₀):

- $TERG'_0 = N'_0 / NB_0 * 100 = 34332 / 328945 * 100 = 10,43\%$

Donc le taux de précision est 89,57%

m. L'image globale:

- Nombre de points colorés:

- NB = 845461 points.

- Taux de perte global:

- $TERI = (N'_0 + N'_1 + N'_2 + N'_3 + N'_4 + N'_5 + N'_6 + N'_7 + N'_8 + N'_9 + N'_{10} + N'_{11}) / NB = 12,75\%$

Donc l'image est reproduite avec une précision de **87,25%**.

b) Image 2 ($\alpha=160$, $d=5$):

- Nombre de contours $N_c = 285$.

1. Temps de reconstruction:

Les temps obtenus dans les différentes expériences sont indiqués dans le tableau suivant:

N° Expérience	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Temps (ms)	170	170	220	170	160	220	160	170	170	170	220	220	170	170	160

Le temps moyen de reconstruction est $T_{\text{moy2}} \approx 193 \text{ ms}$

2. Qualité d'image:

a. Région 1:

- Nombre de points dans la région:
 - NB''_1 : 78458 points.
- Nombre de points additionnels et manquants:
 - N''_1 : $NBP^+ + NBP^- = 1635 + 473 = 2108$.
- Taux de perte ($TERG''_1$):
 - $TERG''_1 = N''_1 / NB_1 * 100 = 2108 / 77296 * 100 = 2,72\%$

Donc le taux de précision est 97,28%

b. Région 2:

- Nombre de points dans la région:
 - NB''_2 : 52620 points dont 47420 en vert, 2924 points en blanc et 2276 points en bleu.
- Nombre de points additionnels et manquants:
 - Région verte (N''_v): $NBP^+ + NBP^- = 1994 + 1513 = 3507$
 - Région blanche (N''_c): $NBP^+ + NBP^- = 322 + 258 = 580$
 - Région bleue (N''_b): $NBP^+ + NBP^- = 255 + 0 = 255$
- Taux de perte ($TERG''_2$):
 - $TERG''_2 = (N''_v + N''_c + N''_b) / NB_2 * 100 = 4342 / 51820 * 100 = 8,38\%$

Donc le taux de précision est 91,62%

c. Région 3:

- Nombre de points dans la région:
 - NB''_3 : 1221 points.
- Nombre de points additionnels et manquants:
 - N''_3 : $NBP^+ + NBP^- = 238 + 66 = 304$
- Taux de perte ($TERG''_3$):
 - $TERG''_3 = N''_3 / NB_3 * 100 = 304 / 1049 * 100 = 28,97\%$

Donc le taux de précision est 71,03%

d. Région 4:

• Nombre de points dans la région:

– NB["]₄: 12437 points.

• Nombre de points additionnels et manquants:

– N["]₄: NBP⁺+NBP⁻ = 1354+509= 1863.

• Taux de perte (TERG["]₄):

– TERG["]₄= N["]₄/NB₄*100 = 1863/11591*100 = 16,07%

Donc le taux de précision est 83,93%

e. Région 5:

• Nombre de points dans la région:

– NB["]₅: 56482 points.

• Nombre de points additionnels et manquants:

– N["]₅: NBP⁺+NBP⁻ = 1707+219= 1926.

• Taux de perte (TERG["]₅):

– TERG["]₅= N["]₅/NB₅*100 = 1926/54994*100 = 3,50%

Donc le taux de précision est 96,50%

f. Région 6:

• Nombre de points dans la région:

– NB["]₆: 12814 points.

• Nombre de points additionnels et manquants:

– N["]₆: NBP⁺+NBP⁻ = 1397+797 = 2194.

• Taux de perte (TERG["]₆):

– TERG["]₆= N["]₆/NB₆*100 = 2194/12214*100 = 17,96%

Donc le taux de précision est 82,04%

g. Région 7:

• Nombre de points dans la région:

– NB["]₇: 3556 points.

• Nombre de points additionnels et manquants:

– N["]₇: NBP⁺+NBP⁻ = 242+10 = 252.

• Taux de perte (TERG["]₇):

– TERG["]₇= N["]₇/NB₇*100 = 252/3324*100 = 7,58%

Donc le taux de précision est 92,41%

h. Région 8:

• Nombre de points dans la région:

– NB["]₈: 2116 points.

• Nombre de points additionnels et manquants:

– N["]₈: NBP⁺+NBP⁻ = 184+71 = 255.

- Taux de perte (TERG"₈):

- $TERG''_8 = N''_8 / NB_8 * 100 = 255 / 2003 * 100 = 12,73\%$

Donc le taux de précision est 87,27%

i. Région 9:

- Nombre de points dans la région:

- $NB''_9: 4961$ points.

- Nombre de points additionnels et manquants:

- $N''_9: NBP^+ + NBP^- = 864 + 354 = 1218$.

- Taux de perte (TERG"₉):

- $TERG''_9 = N''_9 / NB_9 * 100 = 1218 / 4451 * 100 = 27,36\%$

Donc le taux de précision est 72,64%

j. Région 10:

- Nombre de points dans la région:

- $NB''_{10}: 34705$ points.

- Nombre de points additionnels et manquants:

- $N''_{10}: NBP^+ + NBP^- = 1361 + 721 = 2082$.

- Taux de perte (TERG"₁₀):

- $TERG''_{10} = N''_{10} / NB_{10} * 100 = 2082 / 34065 * 100 = 6,11\%$

Donc le taux de précision est 93,89%

k. Région 11:

- Nombre de points dans la région:

- $NB''_{11}: 6816$ points.

- Nombre de points additionnels et manquants:

- $N''_{11}: NBP^+ + NBP^- = 605 + 158 = 763$.

- Taux de perte (TERG"₁₁):

- $TERG''_{11} = N''_{11} / NB_{11} * 100 = 763 / 6369 * 100 = 11,97\%$

Donc le taux de précision est 88,03%

l. Arrière plan:

- Nombre de points dans la région:

- $NB''_0: 322563$ points.

- Nombre de points additionnels et manquants:

- $N''_0: NBP^+ + NBP^- = 5193 + 11575 = 16768$

- Taux de perte (TERG"₀):

- $TERG''_0 = N''_0 / NB_0 * 100 = 16768 / 328945 * 100 = 5,10\%$

Donc le taux de précision est 94,90%

m. L'image globale:

- Nombre de points colorés:

- $NB = 854936$ points.

• Taux de perte global:

$$- \quad TERI = (N''_0 + N''_1 + N''_2 + N''_3 + N''_4 + N''_5 + N''_6 + N''_7 + N''_8 + N''_9 + N''_{10} + N''_{11}) / NB = \mathbf{5.79\%}$$

Donc l'image est reproduite avec une précision de **94,21 %**.

3) Comparaison:

Image reconstruite	Nombre de contours	Nombre de points colorés	Temps moyen de reconstruction (ms)	Qualité
Image(1) ($\alpha=150$, $d=10$)	88	845461	180	87,25%
Image(2) ($\alpha=160$, $d=5$)	285	854936	193	94,21%

On remarque que le temps de reconstruction de la deuxième image est légèrement plus élevé que celui de la première. En ce qui concerne la qualité d'images reproduites, la différence de celle-ci est plus évidente, du fait qu'elle s'est élevée de 87,25% dans la première image à 94,21% dans la deuxième.

Cette différence dans le temps de reconstruction d'images et leurs qualités s'explique par l'augmentation du nombre de contours dans la deuxième image par rapport à la première.

III.3. Discussion des résultats:

En examinant les résultats obtenus dans les différents exemples étudiés ci-dessus on remarque les points suivants:

- Le temps de reconstruction de la deuxième image dans chaque exemple est toujours supérieur à celui de la première (sauf pour la silhouette polygonale). Ceci est dû au nombre de contours dans la deuxième image qui est toujours plus élevé que dans la première. En effet même si cette différence reste très petite (presque nulle) ceci explique que le temps de reconstruction d'une image augmente avec l'augmentation du nombre de contours dans cette dernière.
- La qualité de la deuxième image pour chaque exemple est toujours plus meilleur que dans la première. En effet ceci explique la relation entre la qualité de l'image produite et le nombre de contours dans celle-ci.
- La comparaison entre le temps moyen de reconstruction entre les différents exemples nous permet de déduire que celui-ci est en relation aussi avec le nombre de points colorés dans une image.

Ces résultats nous permettent donc de prouver les hypothèses que nous avons présentées dans le chapitre précédent:

- "Le temps de reconstruction est en relation avec le nombre de contour et le nombre de points à colorer dans l'ensemble des régions."
- "La qualité d'image est en relation avec le nombre de contours dans une description."

IV. Conclusion:

Dans ce chapitre nous avons présenté les résultats obtenus dans la reconstruction des images en utilisant l'application développée.

Pour diversifier les résultats obtenus nous avons choisi trois types d'images: des silhouettes, des images de synthèse et des images réelles. Dans chacun de ces types nous avons choisi deux images (une image simple et l'autre plus compliquée) dont nous avons reconstruit chacune des images à partir de deux descriptions différentes (en changeant les paramètres ' α ' et ' d ' de l'algorithme de Chetverikov).

Dans chaque exemple d'image étudiée nous avons fait une comparaison des résultats obtenus que ce soit en ce qui concerne le temps de reconstruction ou la qualité d'image obtenue. En effet en augmentant le nombre de contour dans la description d'une image, la qualité de l'image reconstruite devient de plus en plus proche de l'image originale surtout si cette dernière contient plus de contour courbés.

Quant au temps de reconstruction de l'image lui aussi augmente en augmentant le nombre de contours dans la description. Celui-ci est en relation avec le nombre de contour d'une part et le nombre de points dans l'image d'une autre part. En effet cette augmentation ne pose aucun problème du fait que celui-ci reste très court (généralement moins d'une seconde) ce qui permet d'utiliser cette application pour la reconstruction de n'importe quelle image quelque soit sa difficulté.

Cette application a prouvé ses aptitudes à reproduire des images simples ou des images réelles contenant des couleurs contiguës comme le cas des Logos. Par contre en ce qui concerne les images réelles complexes contenant des dégradés de couleurs, l'image reconstruite contient des régions colorées avec une seule couleur. En effet ceci est dû au descripteur LWDI qui ne définit qu'une seule couleur par région (couleur de moyenne).

CONCLUSION GENERALE

L'image joue un rôle extrêmement important dans notre vie actuelle, ceci explique la multitude de formats d'image qui contribuent au codage et à l'utilisation de l'image numérique dans différents domaines. Certains formats sont plus utilisés que d'autres ceci revient d'une part aux caractéristiques différentes d'un format à l'autre et d'autre part aux différents domaines qui peuvent requiert à l'utilisation d'un type de format spécifique.

Dans ce travail nous avons parlé d'un nouveau format d'image développé au niveau du laboratoire LRIA de l'USTHB, et basé sur la description de l'image par le biais d'un langage textuel inspirés du langage XML et appelé LWDI.

Ce format se base sur la répartition de l'image en différentes régions de la même couleur et leur décrire par leurs couleurs et contours externes. Ceci à l'avantage d'une part d'offrir un gain souvent considérable dans le volume de l'image par rapport aux formats très répandus (BMP, JPG, GIF,..) et de l'autre part de la possibilité d'appliquer une indexation sur le contenu d'une image ce qui lui permet d'être très utile dans le domaine de la recherche sémantique d'image.

Nous avons proposé dans notre travail une méthode pour la visualisation d'une image à partir de sa description LWDI. Un outil de reconstruction d'image à partir de sa description implémentant cette méthode a été développé sous Delphi et fournit en pièce jointe avec ceci. Dans cette méthode nous avons pris en considération uniquement des images formées des régions dont le rectangle d'aire minimum englobant la silhouette de chacune est en position verticale.

Nous avons proposé aussi une technique pour la quantification des résultats obtenus en ce qui concerne l'efficacité de l'algorithme et la qualité d'image obtenue.

Nous avons illustré notre travail par les résultats obtenus de l'application de cet outil sur différentes images de synthèse et réelles où nous avons montré que la précision avec laquelle les images sont reconstruites dépend de la précision du descripteur (paramètres de l'algorithme de détection de points de courbure).

Nous avons montré aussi que cette technique est plus efficace sur les images contenant des contours réguliers (Logo) que sur des images réelles surtout celles contenant des dégradés de couleur ou des couleurs multiples dans la même région.

Nous prévoyons comme perspectives pour notre travail :

- Le développement d'une technique d'indexation d'objet en se basant sur sa description LWDI pour pouvoir intégrer ceci dans le Web.
- La recherche d'une technique pour la description des régions contenant de multiples ou des dégradés de couleurs, pour pouvoir élargir l'utilisation de cette technique sur des images réelles complexes.

BIBLIOGRAPHIE:

- [ABA 01] Patrick ABATI « Les fichiers d'image ». Cours d'informatique, Académie d'Aix-Marseille.
<http://stielec.ac-aix-marseille.fr/cours> 07/11/2001
- [ACH 05] Tinku Acharya et Ping-Sing Tsai « *JPEG2000 Standard For Image Compression: Concepts, Algorithms And VLSI Architectures* ». Editeur Wiley-Interscience, Janvier 2005.
- [ADO 92] Adobe developers association « TIFF 6.0 révision » Juin 1992
<http://www.adobe.com>
- [ALD 00] Aldon M.J, Borges G.A. « *A split-and-merge segmentation algorithm for line extraction in 2D range images* ». 15^{ème} conférence internationale sur la reconnaissance de formes. Barcelone 2000.
- [AUJ 04] Jean-François Aujol, Basarab Matei. « *Structure and texture compression* ». Rapport de recherche thème3, (INRIA) janvier 2004
- [BAR 06] Mauro Barni « *Document And Image Compression* ». Editeur CRC Press, Mai 2006.
- [BER 01] Bernd Girod et Thomas Wiegand. « *Multi-Frame Motion-Compensated Prediction for Video Transmission* ». Editeur Springer, septembre 2001.
- [BOU 04] Boukelif Aoued « Techniques de compression d'images ». Edition : OPU 2004
- [BRE 03] Stéphane Bres, Jean-Michel Jolion- Frank Lebourgeois « Traitement et analyse des images numérique ». Lavoisier 2003
- [BRO 03] Adrian Brown « Guide d'information sur la conservation numérique, la compression d'image ». Référence du document : DPGN-05 9 juillet 2003.
- [BUR 01] Gilles Burel « Introduction au traitement d'images ». Lavoisier 2001
- [CHE 03] D. Chetverikov. « *A Simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves* », 10th International Conference, CAIP 2003.
- [CLE 99] Charles Clermont « Les fichiers images et le Web ». Cours d'informatique Université de Montréal Canada.
<http://www-1ercycle.med.umontreal.ca> .Février 1999
- [COM 87] CompuServe Incorporated « *A standard defining a mechanism for the storage and transmission of raster-based graphics information* » Juin 1987.

- [DAH 06] Dahmane Afifa « Visualisation des silhouettes à partir de leurs descripteurs XML ». Thèse de magister USTHB 2006.
- [DAV 01] David S. Taubman, Michael Marcellin, «*Jpeg2000: Image Compression Fundamentals, Standards, and Practice* ». Editeur Kluwer Academic Publishers, novembre 2001.
- [EHA 92] Eric Hamilton « *JPEG File Interchange Format Version 1.02* » C-Cube Microsystems, Septembre 1992.
- [ENC 05] Encyclopédie d'informatique libre. «Les formats d'images ». 2005.
<http://www.commentcamarche.net>
- [ESK 95] A. M. ESKICIOGLU et P. S. FISHER, «*Image quality measures and their performance*», IEEE Trans. on Comm., Vol. 43, No. 12, December 1995.
- [FAL 95] FALLAH R. K. et Bolon P. «Une technique d'intégration des résultats de segmentation d'images». Acte du colloque GRETSI, Juan-les-Pins, septembre 1995.
- [GIO 96] Giordan N. «Segmentation non supervisée d'images multispétrales par chaînes de Markov cachées ». Thèse de doctorat, Université technologique de Compiègne, décembre 1996.
- [GUI 96] Guillois Jean Paul «Techniques de compression des images» Edition : Hermes 1996
- [HAM 03] Hammache Tahar & Bazoune Youcef «Segmentation d'image couleur par approche SPLIT & MERGE» Mémoire de PFE (USTHB) 2003
- [HEN 03] Henri Maître « Le traitement des images ». Lavoisier 2003
- [HOW 01] Howard Besser, Diane Bisom, Bernie Hurley, Greg Janee, John Kunze, Reagan Moore, Chaitanya Baru, Terry Ryan, David Walker. «*Digital Image Format Standards*» Mis à jour annuellement 9 juillet 2001 *California Digital Library*.
- [JER 01] Jérôme Morlon «Les techniques de compression: l'image JPEG» <http://developpeur.journaldunet.com> août 2001
- [JOH 99] John Miano «*Compressed Image File Formats: Jpeg, Png, Gif, Xbm, Bmp*» . Editeur Addison-Wesley Professional, septembre 1999.
- [JOL 05] Jean-Michel JOLION «Reconnaissance de Formes et Vision (Projet TELESUN INSA Lyon)» <http://telesun.insa-lyon.fr>. 2005
- [JUN 00] M. JUNG, D. LEGER, M. GAZALET, «Méthode d'évaluation univariante de la qualité des images. Application à des images comprimées selon la norme JPEG», Revue de Traitement du signal, Volume 17 numéro 2, 2000
- [KEE 90] KEEFE J. «*The image as a document* ». Descriptive programs at Renssler Library trends vol 38 .1990

- [KER 97] Kermad C. «Segmentation d'images: recherche d'une mise en œuvre automatique par coopération de méthodes ». Thèse de doctorat, université de Rennes-I 1997
- [KIV 96] Kivi Shapiro «Formats de fichiers graphiques et de fichiers sonores ». Service de technologie de l'information. Bibliothèque Nationale de Canada. <http://www.collectionscanada.ca>. 31 janvier 1996.
- [LAR 03] Slimane LARABI, Saliha BOUAGAR. « *An XML Language for Writing Descriptors of Silhouettes* ». LRIA USTHB conférence SCIA 2003.
- [LAR 07] S. Larabi, «*Textual description of images*», *International Journal of Simulation Modelling*, Vol.6, No. 2, Mai 2007.
- [LAV 06] Mathieu Lavant «*Flash 8 Professional* ». Edition Eyrolles, avril 2006.
- [LEC 06] JP Leclercq «Numériser des images ». Cours d'informatique, Académie de Versailles. <http://www.ac-versailles.fr>: Mis à jour le 18 février 2006.
- [LUC 00] Luciano Volcan Agostini et Sergio Bampi. «*Design of an Integrated Architecture for a JPEG Image Compressing Chip*». *SIM 2000 - XV Microelectronics Seminar, 2000, Torres - Brazil*.
- [MAR 03] Markus Jatton «Compression de données avec perte », Cours de téléinformatique, Institut de Télécommunication. 2003. <http://www.tcom.ch>
- [MAC 04] Macromedia, Incorporated «*Macromedia Flash File Format (SWF)* » 2004. <http://www.half-serious.com/swf/format>
- [MIN 00] Ming-Ting Sun et Amy R. Reibman «*Compressed Video over Networks* ». Marcel Dekker incorporation, septembre 2000.
- [MOK 92] F. Mokhtarian, A. K. Mackworth. «*A theory of multiscale, curvature-based shape representation for planar curves*», *IEEE PAMI*, Vol 14. Août 1992
- [NEL 93] Mark Nelson (Traduit par Hervé Soulard) «La compression de données textes, images, son ». Edition DUNOD 1993.
- [NGA 99] King N. Ngan, K. N. Ngan, et D. Chai «*Advanced Video Coding: Principles and Techniques*». Elsevier science publishers, août 1999.
- [NIC 03] Nicolas Dumoulin «Compression de séquences vidéo et choix des images de références». Rapport de stage de DEA Informatique, Université de Rennes1 – IFSIC Juin 2003.

- [OLS 97] S. OLSSON, M. STROPPIANA, J. BAÑA, «*Objective methods for assessment of video quality: state of the art*», IEEE Transaction on Broadcasting, Vol. 43, No 4, December 1997.
- [OUN 06] OUNI Sonia, «Evaluation de la qualité des images couleurs», thèse de doctorat en cours débutée en 2006.
Laboratoire LERI/CRéSTIC, Université de Reims.
- [PEN 92] William B. Pennebaker et Joan L. Mitchell «*JPEG: Still Image Data Compression Standard* ». Kluwer Academic Publishers, décembre 1992.
- [PIE 99] Alain Pierroux. «Prototype de scanning/compression et archivage de documents». Rapport T06 Projet VIRLIB 1999.
- [ROS 99] Rosenberg C. «Mise en œuvre d'un système adaptatif de segmentation d'image ».Thèse de doctorat, université de Rennes-I, 1999
- [SED 02] Florence Sèdes « *BD multimédia* ». Lavoisier 2002
- [SOH 03] Jean-Claude SOHM «Images et graphiques pour le web »
- CERIG / EFPG décembre 2003.
- [VIA 00] Jean Emmanuel Viallet, Daniel Collobert. (France Télécoms R&D). «Indexation d'images et classification de photographies et de graphismes». Journées CORESA 2000, Poitiers France. 19-20 octobre 2000.
- [WAY 99] Peter Wayner «*Compression Algorithms for Real Programmers* ». Morgan Kaufmann edition, octobre 1999.
- [WHI 99] Richard L. White (STScI), Perry Greenfield (STScI), William Pence (NASA/GSFC) et Doug Tody (NOAO). « *Specifications for Storing Compressed Images in FITS Binary Tables* ». Document décrivant la convention sur la compression des images en n-dimensions. Octobre 1999.
- [WIT 99] I. H. Witten, et Alistair Moffat «*Managing Gigabytes: Compressing and Indexing Documents and Images* ». Editeur Morgan Kaufmann Publishers , mai 1999.
- [YIN 05] Yining Deng, B.S. Manjunath «*JSEG - Segmentation of color-texture regions in images and video* ». "Image Processing & Vision Research Lab" Université de Californie 07/27/2005 <http://vision.ece.ucsb.edu>
- [YUN 99] Yun Q. Shi et Huifang Sun «*Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards* ». Editeur CRC Press, décembre 1999.
- [ZHA 03] D. S. Zhang et G. Lu. «*Review of Shape Representation and Description Techniques*». Pattern Recognition, Editeur Gallimard jeunesse 2003.