Formalisation et Opérationnalisation de Connaissances Graphiques pour l'Interopérabilité en Analyse d'Image de Document.

DEA Perception et Traitement de l'Information

Laboratoire PSI – Université de ROUEN

Septembre 2004

Présenté par : Sébastien PERIN

Stage encadré par M. Eric TRUPIN

Remerciements.

Je tiens à remercier toutes les personnes qui ont permis que ce stage se déroule de la meilleure façon qui soit.

Plus précisément tous mes remerciements vont à M. Eric Trupin qui m'a encadré et offert de bonnes conditions de travail.

Je remercie tout particulièrement M. Mathieu Delalandre qui a contribué à la mise en place de ce stage, pour sa collaboration, son soutien, et le temps qu'il m'a consacré.

Je remercie aussi toutes les personnes qui de près ou de loin m'ont suivi durant mon projet, notamment : M. Sébastien Adam, M. Eugen Barbu, M. Pierre Héroux, M. Jacques Labiche, M. Hervé Locteau, et M. Youssouf Saidali.

Merci à Mustapha pour sa collaboration au projet, pour l'implémentation, et à sa bonne humeur pendant le stage.

Merci aussi à Laurence et Dominique pour leur gentillesse et les plantes vertes dans la salle des stagiaires.

Je tenais à saluer Alain, Carl, Mohamed, Néjib, Yousri, et Zied, pour la bonne ambiance pendant le stage.

Et pour finir, un grand MERCI à Sébastien Leboucher, mon *binôme* de toujours pour les mails de soutien (mutuel) pendant nos stages.

SOMMAIRE

INTRODUCTION. **5**

CHAPITRE 1. ETUDE BIBLIOGRAPHIQUE. **7**

1.1. LES OPERATEURS D'EXTRACTION DE PRIMITIVES GRAPHIQUES. 7
1.1.1. Introduction. *7*
1.1.2. Les techniques d'extraction. *7*
1.1.2.1. Les approches basées sur le squelette – contour. 7
1.1.2.2. Les approches basées sur le suivi de trait. 9
1.1.2.3. Les approches à base de plages – régions. 9
1.1.2.4. Les approches à base de décomposition en mailles. 11
1.1.2.5. Les approches à base de segmentation d'objets. 11
1.1.2.6. Corrections structurelles. 12
1.1.2.7. Simplification d'objets. 14
1.1.3. Conclusion sur l'extraction de primitives graphiques. *15*
1.2. MODELISATION DES CONNAISSANCES GRAPHIQUES EN ANALYSE DE DOCUMENTS. 16
1.2.1. Introduction. *16*
1.2.2. Les systèmes à base de connaissances. *16*
1.2.2.1. Le concept de connaissances. 16
1.2.2.2. Conclusion sur les systèmes à base de connaissances. 18
1.2.3. Ingénierie des connaissances. *19*
1.2.3.1. Modèle conceptuel des connaissances du domaine 20
1.2.3.2. Créer une ontologie. 20
1.2.4. Les formalismes utilisés en analyse de documents graphiques. *21*
1.2.4.1. Les systèmes à base de règles 21
1.2.4.2. Les systèmes à base de frame. 22
1.2.4.3. Les systèmes à base de graphes. 23
1.2.4.4. Les systèmes à base d'un formalisme orienté : données. 25
1.2.5. Les représentations. *25*
1.2.5.1. Les formats images 25
1.2.5.2. Les formats à base de vecteurs. 26
1.2.5.3. Les formats incluant de la sémantique. 27
1.2.6. Conclusion sur les formalismes et représentations. *28*

CHAPITRE 2. UN FORMALISME ET SON OPERATIONNALISATION POUR LES CONNAISSANCES GRAPHIQUES. **30**

2.1. LE PROCESSUS DE MODELISATION 30
2.1.1. La construction du modèle *30*
2.1.2. La méthodologie. *31*
2.1.3. Conclusion sur le processus de modélisation. *32*
2.2. LE FORMALISME ADOPTE. 32
2.2.1. Le formalisme orienté sur les données. *32*
2.2.2. Le formalisme de structuration. *34*
2.2.2.1. La structure liste. 34
2.2.2.2. La structure graphe. 34
2.2.2.3. Conclusion sur le formalisme de structuration. 36
2.2.3. Les objets et attributs standards. *36*
2.2.3.1. Les objets standards. 36
2.2.3.2. Les attributs standards. 37
2.2.4. Mécanismes liés au formalisme. *38*
2.2.4.1. Mécanisme d'externalisation des connaissances. 38

2.2.4.2. Mécanisme de requêtes. 39
2.2.5. Exemple de l'échange des connaissances. *40*

CHAPITRE 3. IMPLEMENTATION DE QUELQUES OPERATEURS POUR L'EXTRACTION DE PRIMITIVES GRAPHIQUES..... 42

3.1. INTRODUCTION. 42
3.2. POLYGONISATION. 42
3.2.1. Méthode de la corde. *42*
3.2.2. Méthode du Split & Merge. *43*
3.2.3. Méthodes basées sur l'aire entre courbe et approximation. *44*
3.3. APPARIEMENT DE CONTOUR. 44
3.3.1. Appariement des vecteurs opposés. *44*
3.3.1.1. Le test d'appariement. 45
3.3.1.2. Le calcul des critères 45
3.3.2. Structuration des quadrilatères construits *45*
3.3.3. Conclusion *46*
3.4. APPROXIMATION DES COURBES. 46
3.4.1. Les méthodes d'approximations d'arcs. *47*
3.4.1.1. Méthodes de D. ELLIMAN. 47
3.4.1.2. Méthodes de X. HILAIRE. 48
3.4.1.3. Méthode proposée par H. LOCTEAU. 48
3.4.1.4. Transformée de Hough locale. 49
3.4.2. Les méthodes d'approximations de courbes. *50*
3.4.2.1. Méthodes de P.L. Rosin et G. West. 50
3.4.2.2. Méthodes de P. L. Rosin. 51
3.4.2.3. Méthodes d'interpolation par des courbes de Bézier cubiques. 52

CHAPITRE 4. MISE EN ŒUVRE DE L'INTEROPERABILITE ENTRE OPERATEURS D'EXTRACTION. 54

4.1. MISE EN ŒUVRE BASEE SUR L'APPROCHE SQUELETTE. 54
4.2. MISE EN ŒUVRE BASEE SUR L'APPROCHE CONTOUR. 55
4.3. CONCLUSIONS ET PERSPECTIVES. 57
4.3.1. Conclusions *57*
4.3.2. Perspectives : *57*

REFERENCES 58

INTRODUCTION.

Dans le vaste domaine de l'interprétation de documents, l'analyse de graphiques est utilisée pour différentes applications, dans des domaines tels que l'interprétation de documents techniques (schémas électriques, plans architecturaux, plans cadastraux, etc.), de la reconnaissance de l'écriture manuscrite (plus spécialement les caractères chinois). L'analyse de graphiques est un domaine pour lequel il existe des applications commerciales éprouvées. Un système d'analyse de graphiques peut se décomposer en deux parties : Une partie traitement (pour l'extraction de primitives graphiques), et la partie système qui gère la partie traitement.

L'extraction de primitives graphiques extrait de la source document "image", les données sous forme vectorielle. Ces données vectorielles sont composées d'objets mathématiques de base associés selon une certaine sémantique : points, lignes, cercles, et courbes. L'analyse de graphique est un processus complexe qui peut faire intervenir différentes méthodes d'extraction de primitives graphiques. Cependant, il existe des points communs entre toutes ces méthodes, ce qui fait que le processus global d'analyse de graphiques peut être vu comme l'enchaînement de différents opérateurs « granulaires ».

La partie système utilise différentes approches dans le but de superviser le processus d'extraction de primitives graphiques. Ces approches appartiennent aux domaines de la reconnaissance de formes et de l'intelligence artificielle. Comme de nombreux systèmes d'interprétation de documents, les systèmes d'analyse de graphiques utilisent des approches à base de connaissances afin de formaliser la description des objets graphiques qu'ils véhiculent.

Bien que ces formalismes soient couramment utilisés dans ces systèmes, les connaissances formalisées sont souvent internes aux traitements. De plus, les systèmes utilisent souvent une modélisation dédiée basée sur l'utilisation d'un formalisme donné. Ces différents aspects rendent les échanges des connaissances graphiques entre systèmes d'analyse de graphiques difficiles, alors que ces systèmes présentent de nombreuses similitudes aux niveaux des données et des algorithmes utilisés. L'utilisation d'une approche basée sur la définition et l'opérationnalisation d'un formalisme commun pour les objets graphiques, permettant différentes modélisations et échanges entre opérateurs, permettrait une gestion optimale et indépendante des enchaînements entre les traitements.

Cette étude se propose d'apporter une contribution au domaine de la formalisation, de la modélisation, et de l'opérationnalisation des connaissances graphiques dans les systèmes d'analyse de graphiques. Ce rapport se décompose en trois parties :

1. L'étude bibliographique des différentes approches d'extraction de primitives graphiques.
2. La conception d'une approche pour la formalisation et l'opérationnalisation des connaissances graphiques, permettant l'utilisation de différentes modélisations, et assurant ainsi l'interopérabilité entre opérateurs d'extraction de primitives graphiques.
3. La réalisation de différents opérateurs d'extraction de primitives graphiques, basés sur l'utilisation de notre approche

Pour conclure, un exemple illustrera la mise en œuvre de l'interopérabilité entre opérateurs d'extraction de primitives graphiques.

Chapitre 1 : Etude bibliographique

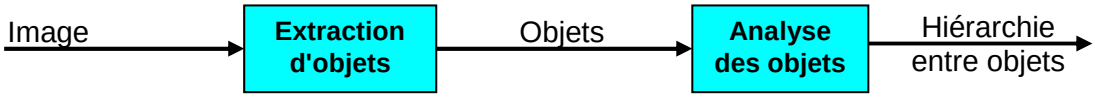
Chapitre 1. Etude bibliographique.

1.1. Les opérateurs d'extraction de primitives graphiques.

1.1.1. Introduction.

Les traitements effectués pour l'extraction de primitives graphiques sont la succession de deux processus :

- Les techniques d'extraction d'objets de bas niveau structurel.
- L'analyse des objets.



Il ne s'agit pas de faire de cette étude une liste exhaustive de tous les traitements possibles, mais il faut y voir un "florilège" de méthodes types.

1.1.2. Les techniques d'extraction.

Les techniques d'extraction d'objets structurés décomposent les composantes connexes de l'image en graphes d'objets.

Il existe cinq approches différentes, les méthodes à base de :

- Squelettisation - détection de contours,
- Suivi de trait,
- Décomposition en plages - régions,
- Décomposition en mailles,
- Segmentation d'objets.

1.1.2.1. Les approches basées sur le squelette – contour.

Les méthodes à base de squelette - contour sont les plus utilisées. Elles ont en commun leur décomposition en deux parties.

Ainsi la squelettisation extrait dans un premier temps les images de squelette [Lam95]. On peut obtenir le squelette par deux moyens :

- par amincissements successifs
- par transformées de distances [Ablameyko00]

Puis dans un second temps, les images de squelette sont analysées afin d'en extraire des graphes de pixels [Tombre99]. Cette analyse utilise différentes méthodes basées, soit sur l'analyse de la connexité [Tombre99], soit sur la destruction des pixels 3-connexes et plus [Delalandre03a].

La Figure 1 donne un exemple de squelette résultant de l'extraction.

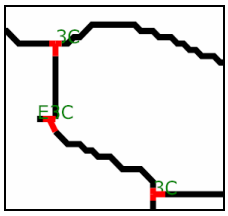


Figure 1 : Squelette avec marquage des pixels 3-connexes.

Voyons maintenant l'extraction ou la détection[1] de contours. Il existe deux catégories de méthodes :

- Celles utilisant des images intermédiaires des contours.
  Elles sont basées sur la morphologie mathématique ou le test de voisinages [Delalandre03a].
  Puis elles sont suivies d'une étape d'extraction des chaînes de pixels représentant les contours.
- Celles qui extraient directement les chaînes de pixels des contours de l'image initiale.
  Ces méthodes utilisent le suivi de contour [Ablameyko00], ou le marquage de composantes connexes [Delalandre03a].
  On peut ainsi exprimer des relations d'inclusions entre les chaînes, et sélectionner les contours internes et/ou externes aux formes [Delalandre03a].

La Figure 2 est un exemple de marquage sur les chaînes de contours extérieurs.

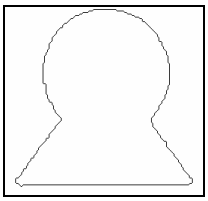


Figure 2 : Marquage sur contours extérieurs

On l'obtient par traitement de la Figure 3 provenant de [Delalandre03a].

Figure 3 : Image originale.

[1] Termes utilisés de façon indifférente par les auteurs

1.1.2.2. Les approches basées sur le suivi de trait.

Les méthodes à base de suivi de trait[1] procèdent à une analyse directe des images sans utiliser de représentations intermédiaires.

Le suivit de trait est basé sur l'utilisation d'éléments structurants pour parcourir les formes, de type pixellaire[2] [Wenyin99]. Il existe aussi des types surfaciques [Delalandre03b] (cercle, élément gaussien.)

Ces méthodes produisent des graphes d'objets géométriques (cercles et vecteurs [Song03]), ou de pixels [Wenyin99]. Tout dépend du type de parcours effectué (linéaire [Wenyin99], en zigzag [Wenyin99]), et de la progression séquentielle (pas à pas) ou continue [Wenyin99] de l'élément structurant dans la forme.

Puis intervient la procédure d'analyse qui est subdivisée comme suit :

- l'analyse des lignes,
- l'analyse des jonctions.

Dans les deux cas, l'élément structurant employé peut être de type pixellaire [Wenyin99], de type surfacique [Delalandre03b], ou la méthode peut être hybride [Delalandre03b].

La Figure 4 donne un exemple de parcours en zigzag par un élément structurant de type pixellaire [Wenyin99].

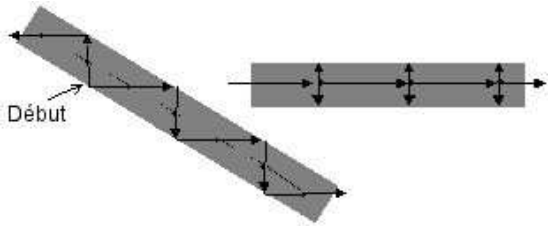


Figure 4 : Suivit de trait : OZZ.

1.1.2.3. Les approches à base de plages – régions.

Les méthodes à base de décomposition en plages sont issues des études de la reconnaissance de symboles et la reconnaissance de l'écriture manuscrite chinoise [Lin02].

Une plage est une séquence de pixels noirs, horizontale ou verticale. En partant de cette définition de Burge datant de 1998 et donnée dans [Delalandre03b], un graphe de plages verticales et horizontales le « Mixed Run Graph » [Wenyin99] [Song03] est construit comme dans la Figure 5.

[1] ou "parcours des formes"
[2] méthode de "sparse-pixel" de Dori

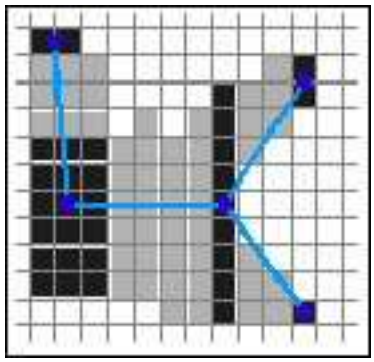


Figure 5 : Plage avec son graphe associé.

Les plages 2-connexes successives sont concaténées par des arcs dans le graphe, et les plages verticales et horizontales juxtaposées sont assimilées à des nœuds jonctions.

Les méthodes à base de décomposition en régions reposent sur la décomposition en composantes connexes d'un ensemble de régions. Y.S. Chen en 1994 et R. Cao en 2000 (cf. [Delalandre03b]) calculent des informations d'orientation pour chacun des pixels de l'image par rapport aux pixels de contour de la forme. Ainsi les pics des directions majoritaires pour chacun des pixels peuvent être recherchés par la suite.

Donc un pixel est étiqueté comme appartenant à une région :

- ligne (R),
- extrémité (E),
- ou jonction (S),

Voir la Figure 6 illustrant cet étiquetage.

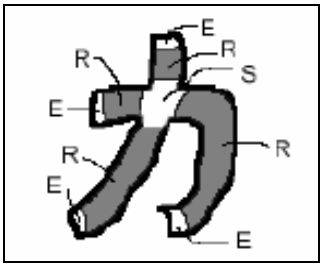


Figure 6 : Régions avec étiquetage.

Ce sont les ruptures ou les blocages des vagues qui déterminent les frontières entre les régions.

Le traitement se termine par une analyse des graphes de régions obtenus afin de construire les régions 'jonction' et 'ligne' des formes.

1.1.2.4. Les approches à base de décomposition en mailles.

Les méthodes à base de décomposition en mailles proposées par Vaxivière en 1995 ne sont actuellement utilisées que pour des applications de vectorisation [Wenyin99]. On superpose un quadrillage sur l'image (voir Figure 7) formant ainsi un pavage découpé : les mailles.

Puis dans une autre phase, les mailles sont reconnues d'après les modèles de mailles [Song03] enregistrés dans une base.

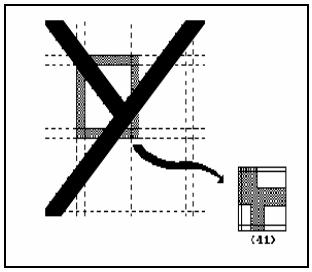


Figure 7 : Maillage avec patron.

La disposition sur l'image des mailles reconnues est analysée afin de construire les relations structurelles entre les mailles.

1.1.2.5. Les approches à base de segmentation d'objets.

Les méthodes à base de segmentation d'objets proposées par Matas en 1999 puis par Su en 2002, consistent à segmenter directement des objets de l'image comme les lignes, les arcs, et les ellipses.

Ces méthodes emploient des transformées mathématiques afin de changer l'espace de représentation des images vers un espace de représentation.

Plusieurs méthodes existent, telles que la transformée de Hough [Wenyin99] [Song03] et les filtres de Gabor.

Cet espace de représentation est exploité afin de rechercher les objets en fonction de leurs modèles mathématiques. Ainsi les lignes diagonales, verticales, et horizontales sont extraites pour la reconnaissance de l'écriture manuscrite chinoise comme dans la Figure 8.

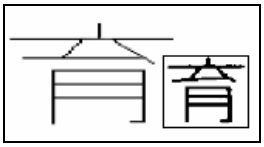


Figure 8 : Segmentation sur l'écriture manuscrite chinoise.

Certains travaux portent sur la segmentation directe des jonctions (jonction en 'T', en 'X', etc.) voir les travaux de Chen en 2000 exposés dans [Song03].

Dans le cas où la méthode employée ne permet pas la segmentation des jonctions, elles peuvent être reconstruites par analyse des croisements géométriques ou par la recherche des extrémités d'objets proches les une des autres.

1.1.2.6. Corrections structurelles.

Les techniques d'extraction produisent malheureusement du bruit. Des méthodes de correction de graphe [AhSoon98], sont proposées.

1.1.2.6.1. Cercle maximum inscrit.

Une squeletisation aboutit parfois à la présence de nœuds redondants. Le critère du cercle maximum [Lin02] peut alors être utilisé de façon à fusionner les nœuds proches et à supprimer les pixels à proximité des zones de jonction (Figure 9.)

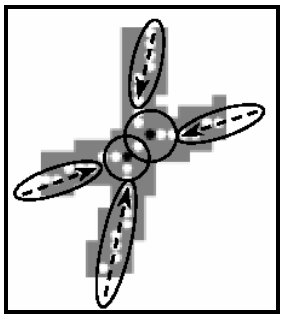


Figure 9 : Cercle maximum inscrit.

1.1.2.6.2. Ebarbulage.

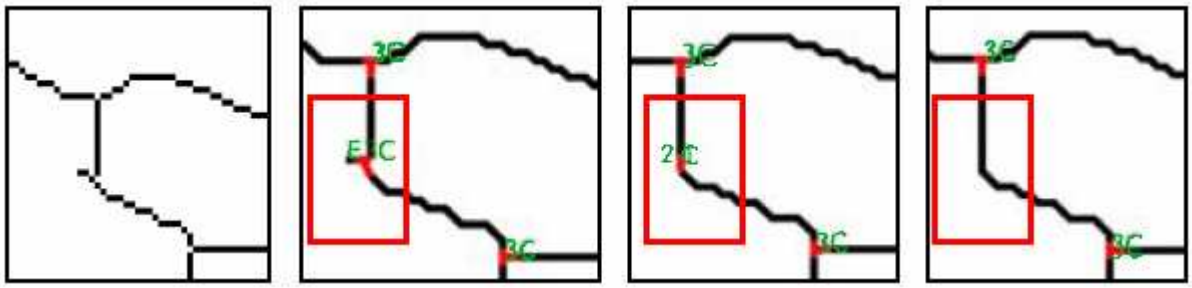
Une étude des méthodes d'ébarbulage est réalisée dans [Ablameyko00].

Une technique simple utilise un seuil fixe en parcourant les chaînes du squelette. Si une chaîne possède une extrémité libre et a une longueur inférieure à ce seuil, la chaîne est considérée comme une barbule et peut être supprimée.

Mais cette méthode est inadaptée aux traits trop épais. De plus les traits trop fins ou trop courts seront supprimés.

Une autre méthode réalise un seuillage adapté au trait analysé. Ainsi une chaîne est supprimée si elle possède une extrémité libre et une autre connectée à une jonction dont sont issues au moins deux chaînes. Si la longueur de cette chaîne est inférieure à l'épaisseur du trait au niveau de la jonction alors la chaîne est une barbule.

La Figure 10 donne un exemple de reconstruction de jonctions (a, b), suivie d'un traitement d'ébarbulage (b, c), puis d'un traitement de fusion des listes (c, d).



a) Squelette. **b) Détection des jonctions.** **c) Ebarbulage.** **d) fusion des listes.**

Figure 10 : Ebarbulage.

A noter que l'ébarbulage réduit la taille du squelette, cette réduction doit être prise en compte pour la polygonalisation.

1.1.2.6.3. Fusion.

La correction de l'information structurelle peut générer du bruit. Ce bruit est fonction de nombreux paramètres, comme la détection de faux nœuds, la fausse détection d'une fin de trait, l'absence de détection de trait.

D. Dori ([Dori99]) propose une étape de correction de l'information structurelle. Cette correction agit sur les listes et les nœuds du graphe.

Ainsi les listes du graphe sont fusionnées par une progression pas à pas dans le trait. Uniquement dans le cas où la variation de l'orientation est très faible entre les pas successifs (Figure 11.)

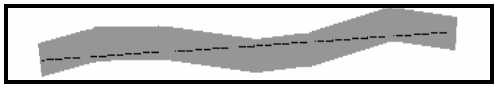


Figure 11 : Fusion des progressions à faibles variations directionnelles.

Les nœuds du graphe peuvent être corrigés de deux manières selon un mécanisme lié à l'intersection des prolongements des suivis :

Soit la correction fusionne les listes fragmentées en une seule liste (Figure 12.)

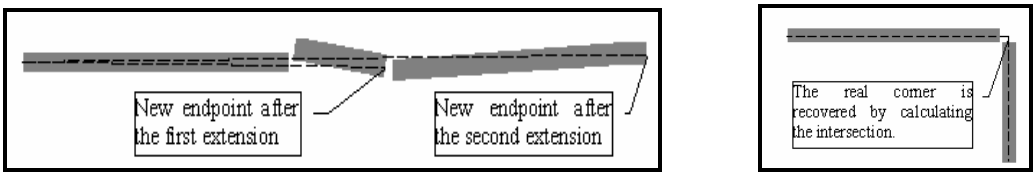


Figure 12 : Fusion de liste fragmentée

Soit elle traite les nœuds 3-connexe ou plus, non détectés. La méthode analyse alors les intersections des prolongements des suivis par rapport aux suivis voisins pour créer les nœuds disparus (Figure 13.)

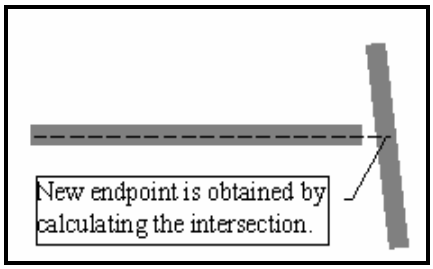


Figure 13 : Reconstruction d'un nœud.

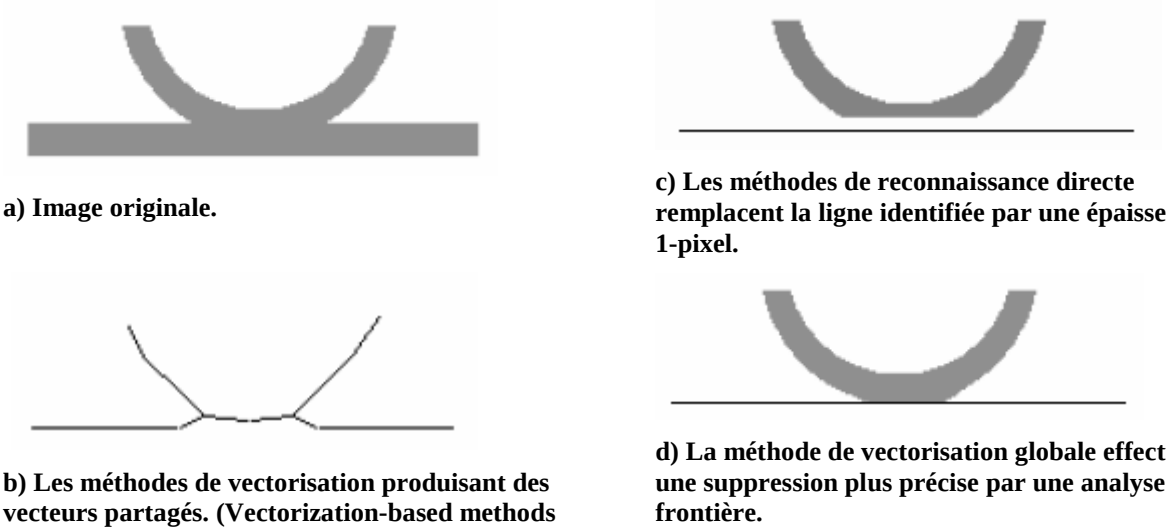
1.1.2.7. Simplification d'objets.

La simplification d'objet permet l'élimination des objets redondants. Ainsi la complexité de la représentation d'une jonction est simplifiée. Cependant, la simplification d'objet peut aboutir à une perte d'information.

Dans [Song03], est donné l'exemple suivant : un arc touche légèrement une ligne droite voir Figure 14a. La simplification d'objet de cette figure entraîne deux problèmes.

La première difficulté est la perte d'objet image par la suppression d'un vecteur partagé Figure 14b avec un objet vectoriel. Alors que le repérage des vecteurs plutôt que leur suppression, peut garder l'intégralité des deux objets, mais la complexité en temps de recherche est très élevée.

La deuxième difficulté vient de la structure des données. La plupart des méthodes de vectorisation structurent les vecteurs en chaine (Figure 14c) leur analyse en est difficile. Cependant, les méthodes d'identification directe Figure 14d ne souffrent pas de telles difficultés.



a) Image originale.

b) Les méthodes de vectorisation produisant des vecteurs partagés. (Vectorization-based methods produce shared vectors)

c) Les méthodes de reconnaissance directe remplacent la ligne identifiée par une épaisseur 1-pixel.

d) La méthode de vectorisation globale effectue une suppression plus précise par une analyse de frontière.

Figure 14 : Exemple de recouvrement partiel et le résultat de différents traitements

(En gris : image; En noir : vecteurs)

Un autre exemple de simplification d'objet est étudiée en [Song02], et permet de réduire la complexité du processus de reconnaissance. Ce traitement permet d'effacer des objets de type arc.

1.1.3. Conclusion sur l'extraction de primitives graphiques.

L'ensemble des méthodes étudiées dans ce chapitre ainsi que les opérateurs d' d'extraction de primitives graphiques (Chapitre 3.) sont récapitulés dans le Tableau 1, en correspondance sont donnés les flux des primitives graphiques en entrée et sortie.

Ces approches sont présentées de façon indépendante dans la littérature, cependant il existe divers points communs entre elles.

		Primitives d'entrée	Primitives de sortie
LES TECHNIQUES D'EXTRACTION			
	Squelette Contour	Image[1]	Point
	Suivit de trait	Image	Point
	Plages/Régions	Image	Surface
	Mailles	Image	Patron de la base
	Segmentation d'objets	Image	Ligne arc ellipse
L'ANALYSE DES OBJETS (Chapitre 3.)			
Polygonisation	La corde	Point	Ligne
	Split & Merge	Point	Ligne
	Critère surfacique	Point	Ligne
Appariement de contour		Ligne	Polygone
Approximation des arcs et des courbes	Elliman	Ligne	Ligne
	Hillaire	Point/ligne	Ligne
	Locteau	Point	Centre, rayon, début, fin, sens de l'arc
	Rosin & West	Point	Graphe[2]
	Rosin	Point	Polygone
	TH[3] locale	Image/point	Centre, rayon du cercle
	Bézier cubique	Point	Points de contrôle
Correction structurelle	MIC	Ligne	Ligne
	Ebarbulage	Ligne	Ligne
	Fusion	Graphe	Graphe
Simplifications d'objet		Graphe	Graphe

Tableau 1 : Les objets gérés par les différentes méthodes.

En effet ces méthodes utilisent une sémantique basée sur les objets classiques de la géométrie euclidienne : le point, le segment de droite (le terme "*ligne*" est plus couramment utilisé), l'arc de cercle, la courbe.

D'autre part des traitements granulaires sont communs entre ces approches.

Il y aurait donc un intérêt certain à combiner ces méthodes afin de simplifier le problème de l'extraction

[1] Ou encore de tableau de pixels
[2] Il s'agit de graphes relationnels attribués
[3] TH : Transformée de Hough

1.2. Modélisation des connaissances graphiques en analyse de documents.

1.2.1. Introduction.

Dans le chapitre précédent nous avons identifié des besoins en matière de formalisation des connaissances dans les systèmes d'analyse de graphique. Le système à réaliser peut donc être vu comme un système à base de connaissances (SBC) comme cela est présenté dans [Saidali02] et [Haton91]. L'intégration des connaissances est donc impérative lors de la conception de ce dit système. Il est donc utile d'analyser quelques notions de l'ingénierie des connaissances, afin d'en extraire un formalisme adéquat.

La modélisation de la connaissance est une thématique importante dans le domaine de l'intelligence artificielle, elle implique la description de la nature et de la structure de la connaissance pour définir le comportement désiré de résolution des problèmes.

La modélisation repose sur l'ingénierie des connaissances qui est définie par Jean Charlet et Bruno Bachimont [Charlet98] comme suit :

- L'ingénierie des connaissances correspond à l'étude de la raison computationnelle. C'est l'ingénierie des systèmes dynamiques que permet l'ordinateur en fonction des interprétations qu'ils suscitent.

- L'ingénierie des connaissances correspond à l'étude des modèles symboliques formels plongés dans des systèmes d'usage : c'est l'ingénierie informatique et logique de modèles en fonction des usages qu'ils rendent possibles et des appropriations qu'ils permettent.

- L'ingénierie des connaissances correspond à l'étude des concepts, méthodes et techniques permettant de modéliser et/ou d'acquérir les connaissances dans des domaines se formalisant peu ou pas.

La conception du système va tenter de s'appuyer sur ces trois définitions

1.2.2. Les systèmes à base de connaissances.

1.2.2.1. Le concept de connaissances.

Sous ce terme de *connaissance*, interviennent plusieurs associations. Suivant les points de vue, il peut s'agir à la fois de faits (données observées), d'informations structurées, de mécanismes de raisonnement (théories, méthodes, déductions, etc.) ou de mécanismes d'analyse et de synthèse (modélisation, étude de la complexité, etc.)

Par ailleurs, toutes ces connaissances sont reliées, puisqu'il ne peut pas y avoir de théorie sans observation et sans expérience, et réciproquement, ou de transmission de la connaissance sans langage qu'il soit naturel ou artificiel.

La *connaissance* est un concept qui ne peut être définit formellement. Cependant, dans l'état actuel de la recherche, on s'accorde à dire qu'il existe trois niveaux si l'on se place selon le point de vue des techniques informatiques. Dans la mesure où il existe des outils et techniques clairement identifiées pour les manipuler, ces niveaux sont : Données, Information et Connaissances.

1.2.2.1.1. Données, Informations, et Connaissances.

Ce paragraphe va donner une distinction entre :

- les données et,
- les informations,

Les Données sont classiquement vues comme étant des valeurs numériques perçues au travers de capteurs. Il s'agit entre autres de codes, de faits, de sons, d'images ou de textes. Les données sont des entités évoluant et/ou se renouvelant en fonction de leur utilisation. Les systèmes de gestion de bases de données (SGBD) permettent de les accumuler, de les mettre à jour de manière structurée et d'en extraire des sous-ensembles par l'intermédiaire d'un langage de requête.

Les informations sont quand à elles composées par des données encapsulées par un contenu sémantique. La Figure 15 illustre cette encapsulation.

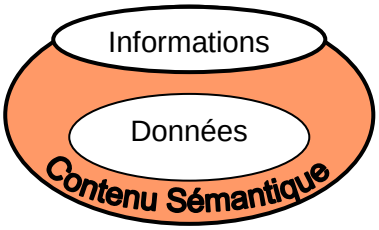


Figure 15 : Informations = Données + contenu sémantique.

Pour conclure, la *connaissance* est une information dans l'action. Les données sont associées à leur contenu sémantique, permettant ainsi de générer des connaissances suivant un contexte d'utilisation dans un système (on parle de système à base de connaissances). Autrement dit les connaissances sont l'encapsulation des informations opérationnalisées via un système cf. Figure 16.

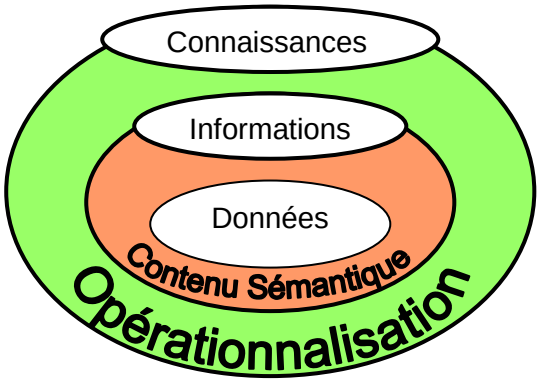


Figure 16 : Connaissances = Informations + opérationnalisation.

1.2.2.1.2. Formalisme et représentation dans les SBC

Les connaissances sont représentées au niveau procédural (interne) et déclaratif (externe) dans les systèmes informatiques [Haton91]. Dans les deux cas, le type de représentation des connaissances est fonction du formalisme de représentation adopté.

Certains formalismes simples s'apparentent plus à de la représentation de données [Lucas86] (matrice, pile, etc.), d'autres plus complexes à de la représentation de connaissance [Kayser97] (logique de description, graphe, règle, objet structuré "frame", etc.). Les systèmes capables de déclarer et d'exploiter des connaissances sont qualifiés de Système à Base de Connaissance (SBC) [Haton91]. Dans ces systèmes, la base de connaissance regroupe et structure de façon externe (déclarative) les connaissances du système [Kayser97]. Pour cela de nombreuses méthodes existent, de l'utilisation de techniques de base de données [Ullman89] ou de langage de représentation des connaissances [Kayser97].

1.2.2.1.3. Les types de connaissances

N. Aussenac-Gilles et J. Charlet présentent dans [Aussenac99] plusieurs types de connaissances que l'on peut différencier:

- **La connaissance dans un environnement technique**. Cette connaissance n'est possible que dans un environnement *technique* qui est le nôtre, qu'il apparaisse fortement technique (l'industrie, le monde professionnel) ou non (le monde quotidien.)
- La connaissance dans le sens de "**l'interprétation humaine**". La connaissance et sa représentation sont les résultats d'une manipulation symbolique effectuée par la machine, de plus ces résultats doivent être validés par les utilisateurs des programmes.
- **La connaissance = *f* (l'outil informatique)**. Dans ce cas, la technique est l'informatique, par ses outils et ses capacités de mémorisation, elle permet alors à l'homme de se constituer des connaissances qui évoluent et s'accumulent mais qui n'existeraient pas sans cette technique.
- **La connaissance impliquant une action**. Ici on considère qu'il y a *connaissance*, si la faculté d'utiliser l'information à bon escient est validée. La connaissance est ainsi liée à l'*action*. Cette action est ici une réécriture.
- **La connaissance = *f* (contexte)**. Où la connaissance existe uniquement grâce à un contexte d'utilisation de l'information.
- **La connaissance = *f* (tâche)**. Dans ce cas, la connaissance *a priori* n'existe pas, elle est construite à partir d'un projet propre au modélisateur. Il faut alors étudier cette construction pour essayer d'en dégager des invariants méthodologiques.

1.2.2.2. Conclusion sur les systèmes à base de connaissances.

La capitalisation et la gestion des connaissances[1] sont en plein essor avec les applications de type : ontologie médicale, Internet, systèmes d'information d'entreprise, etc. Un SBC au sens large n'exploite pas obligatoirement de capacités de raisonnement et n'intègre donc pas nécessairement de connaissances formelles. Par contre, l'objectif d'un SBC axé sur la gestion est de trouver et fournir les connaissances. En effet il doit être capable de les conserver et les valoriser, de pouvoir y accéder et les actualiser, de les diffuser, et de combiner leurs actions.

[1] ou *Knowledge Management*

Le processus de capitalisation des connaissances permet la réutilisation des connaissances stockées préalablement et modélisées dans un domaine donné afin de réaliser de nouvelles tâches.

Ainsi un SBC peut être défini comme se trouvant à la frontière entre un système contenant des connaissances et un système d'information. Un SBC ne résout pas automatiquement une tâche, mais aide plutôt l'utilisateur en lui fournissant des informations et en le laissant libre de leur interprétation et de leur évaluation contextuelle.

Le concept même de système de gestion de connaissances fait intervenir plusieurs réflexions. Celles-ci concernent la linguistique pour dégager la formulation langagière des connaissances, la terminologie et l'ontologie pour étudier les concepts, la logique pour élaborer les modèles formels, l'informatique pour les opérationnaliser, etc.…

La démarche que ces réflexions impliquent, s'inscrit dans la discipline spécifique qu'est l'ingénierie des connaissances.

1.2.3. Ingénierie des connaissances.

L'ingénierie des connaissances propose des concepts, méthodes et techniques permettant d'acquérir, de modéliser et de formaliser des connaissances pour les mobiliser dans l'activité d'une organisation.

N. Aussenac-Gilles et J. Charlet présentent dans [Aussenac99] une liste des différentes tâches dans lesquelles l'ingénierie des connaissances intervient :

- La définition d'une aide pour l'utilisateur (méthodes, outils logiciels ou non, organisation du travail)
- La modélisation des connaissances, individuelles ou collectives (telles que la mémoire d'entreprise [Dieng98]), explicites ou implicites, stabilisées ou évolutives, expertes ou techniques, etc.
- A rendre ces connaissances accessibles sous une forme définie en fonction du contexte, opérationnelle ou non.

La réussite d'une démarche d'ingénierie des connaissances, dépend d'une part d'une modélisation suffisamment générique et faisant sens pour les différents acteurs (ingénieurs cogniticiens, experts métiers, utilisateurs, etc.) Ces acteurs doivent pouvoir s'approprier le comportement du système et interagir avec lui. D'autre part la réussite de la démarche dépend, de l'attention portée à la formalisation des connaissances, puis au passage à la spécification et pour finir à la réalisation du système de traitement de l'information.

D'après Jean Charlet [Charlet02], les travaux en ingénierie des connaissances sont fondés sur la nécessité de modéliser de manière explicite les connaissances. Il s'agit de construire des modèles adaptés à la nature des connaissances à décrire pour pouvoir ensuite les représenter dans des formalismes adéquats. Dans le cadre des SBC, trois types de connaissances font l'objet de modèles distincts avec des primitives de modélisation propres :

- les connaissances du domaine,
- les tâches,
- les méthodes.

L'ensemble de ces modèles forme alors ce qui est communément appelé en ingénierie des connaissances, le *modèle conceptuel*.

L'analyse du domaine et l'analyse des tâches d'un SBC sont à rapprocher des analyses statiques et fonctionnelles effectuées en génie logiciel. En revanche, l'analyse de la dynamique en génie logiciel consiste à étudier les conséquences d'événements extérieurs sur une application alors que pour un SBC, il s'agit de rendre compte du contrôle interne du

comportement du système à implémenter. En effet, le problème de l'efficacité d'un SBC ne peut être posé uniquement en termes d'efficacité algorithmique mais va également se poser par rapport aux connaissances utilisées ; celles-ci devant alors être décrites de manière déclarative en phase d'analyse.

1.2.3.1. Modèle conceptuel des connaissances du domaine

Le modèle conceptuel d'un SBC combine un ou plusieurs modèles de raisonnement, en plus d'une représentation indispensable pour cette modélisation.

D'après [Aussenac99], le modèle conceptuel est la concaténation des *connaissances stratégiques* et des *connaissances du domaine*.

Modèle Conceptuel = Connaissances Stratégiques + Connaissances du domaine

Les auteurs entendent par **connaissances stratégiques**, tout ce qui exprime comment une tâche va être effectuée (elles utilisent des connaissances du domaine.) d'autre part les **connaissances du domaine** sont les concepts manipulés et leurs relations, c'est à dire ce qui est vrai dans le domaine pour la tâche considérée.

Le modèle conceptuel peut être utilisé à différentes fins telles que la validation informelle d'une méthode de résolution des problèmes, ou la maintenance du SBC final, ou encore comme guide pour le recueil de connaissances.

Le modèle conceptuel a pour objectif de réduire la complexité des tâches à exécuter par les systèmes en les représentant d'une façon synthétique, facile à comprendre et permettant de se focaliser sur des aspects importants dans le développement du système. Ce modèle regroupe l'ensemble des concepts du domaine.

Autrement dit, le modèle conceptuel sert à former les entités du domaine et leurs interactions en un concept. Le modèle conceptuel est un intermédiaire entre les connaissances et le code. Il est un langage d'expression des connaissances, partagé entre l'expert et le cogniticien.

Le modèle est une abstraction qui permet de réduire la complexité en se focalisant sur certains aspects, en fonction de certains buts. Mais ce modèle devrait permettre plus : manipuler les objets et interpréter les résultats de la manipulation

Pour conclure le modèle conceptuel explicite la manière dont le système résout les tâches ainsi que les connaissances qu'il utilise.

1.2.3.2. Créer une ontologie.

Les termes du domaine ne sont pas obligatoirement des concepts dans la mesure où ils ne sont pas nécessairement interprétés de la même manière par plusieurs spécialistes. De plus pour modéliser les connaissances d'un domaine quelconque, il est nécessaire de restreindre l'interprétation d'un terme pour qu'il puisse être une primitive.

Van Den Elst dans sa thèse [VanDenElst96] présente la connaissance du domaine comme une composition des :

- définitions d'ontologies,
- mises en correspondance d'ontologies,
- modèles du domaine.

Où une définition d'ontologies contient les limites référant aux catégories qui doivent être distinguées dans un domaine particulier, et les termes qui se rapportent aux relations qui peuvent exister entre ces catégories ou leurs instances.

La mise en correspondance d'ontologies préconise une "traduction" entre les constructions définies dans une ontologie vers une autre.

Le modèle du domaine est constitué des connaissances déclaratives du domaine décrites par les termes de l'ontologie. C'est un ensemble d'états/expressions qui va être employé dans certaines étapes de raisonnement.

En conclusion, une ontologie est l'ensemble des objets reconnus comme existant dans le domaine. On cherche alors à formaliser les concepts pour concevoir une ontologie. De plus, construire une ontologie c'est aussi décider de la manière d'être et d'exister des objets.

1.2.4. Les formalismes utilisés en analyse de documents graphiques.

Un état de l'art dressé dans [Crevier97] montre que dans le domaine de l'interprétation de documents, la plupart des formalismes utilisés en intelligence artificielle peuvent être adoptés. L'enchaînement des formalismes donné dans ce paragraphe suit l'évolution historique des systèmes qui a globalement suivi les grandes tendances de l'Intelligence Artificielle.

Chacun de ces formalismes, sera accompagné d'une analyse de ses avantages et inconvénients.

1.2.4.1. Les systèmes à base de règles.

Avec les premiers résultats de l'Intelligence Artificielle, les premiers systèmes de vision à base de connaissances pouvaient être vus comme des systèmes de production, à base de règles.

Une règle répond au principe suivant :

Faire *Action(s)* si *Condition(s)*

Le premier exemple à avoir adopté ce formalisme des règles est le système développé par l'équipe de Levine ([Levine78]). Les règles y sont utilisées pour représenter les connaissances au niveau de la scène (le domaine dans la terminologie utilisée précédemment). Ces règles floues, qui décrivent le monde, permettent l'utilisation d'une approche descendante, guidée par les buts.

Ce système exploite en outre deux bases de données relationnelles pour représenter les connaissances descriptives.

La première (Mémoire à Long Terme MLT) permet la description a priori des objets à extraire, de leurs propriétés (dont les valeurs moyennes sont stockées) et de leurs relations spatiales.

La seconde (Mémoire à Court Terme MCT) sert quant à elle de mémoire de travail en stockant les résultats de l'interprétation en cours.

Dans ces bases, les données sont hiérarchisées suivant trois niveaux d'abstraction : le bas niveau (image), le niveau intermédiaire (contours/région) et le haut niveau (objets). A chacun des niveaux de la MCT sont présentes les instances du modèle défini dans la MLT, avec des valeurs affectées aux attributs et aux relations spatiales.

Une interprétation associée à chaque région et un historique des traitements effectués sont également intégrés à cette MLT.

Une originalité de ce système est de décrire les objets non seulement par une liste de propriétés, mais aussi en y faisant figurer les propriétés particulières qui permettront de discriminer un objet donné

Le formalisme des règles est également à la base du système ACRONYM de Brooks ([Brooks83]). Ce dernier est l'un des premiers systèmes dont la vocation est d'être générique, en ayant donc pour ambition d'être indépendant du domaine d'application. Toutefois, il n'a concrètement été testé que sur des images d'aéroport.

Dans ce système, les objets sont modélisés en 3-D et détectés sur des images 2-D. Les apparences géométriques des objets dans la scène sont modélisées par l'intermédiaire des primitives les constituant, et de leurs relations spatiales. Elles sont représentées sous forme de 280 règles de production dans un module appelé module de prédiction.

En 3-D, les descriptions de ces objets sont effectuées à l'aide d'expressions qui sont des combinaisons de variables prédéfinies utilisant des opérateurs tels que +, -, *, < et >. Le système fait aussi appel à une modélisation hiérarchique des objets exploitant des liens de type "partie-de" ou "sorte-de". Elle est effectuée en utilisant le formalisme des frames de Minsky (voir § 1.2.4.2) et de cônes généralisés.

Ces systèmes à base de règles ont globalement montré l'intérêt d'une approche à base de connaissances. Toutefois, les systèmes à base de règles sont parfois peu évolutifs, en effet les règles forment un formalisme relativement rigide.

1.2.4.2. Les systèmes à base de frame.

La notion de frame a été introduite par les travaux de Marvin Minsky ([Minsky75]). Une représentation à base de frames est inspirée du fait que la réaction de l'être humain se base sur des représentations génériques de situations auxquelles il a déjà été confronté. En fonction des spécificités de la situation présente, l'Homme affine sa perception de la scène pour réagir de façon plus appropriée. Une frame est un cadre qui correspond à une situation déjà rencontrée.

Elle est composée de slots, qui servent à la caractériser(voir la Figure 17). Ces slots fournissent des informations sur la hiérarchie des frames ("sorte de…", "est un…"), sur leurs propriétés (sa couleur…), sur leur rôle et sur les relations les unissant (près de, au-dessus…).

Ils sont associés à des facettes qui permettent de caractériser la nature et le comportement d'un attribut et qui décrivent par exemple le type et le domaine de validité du slot.

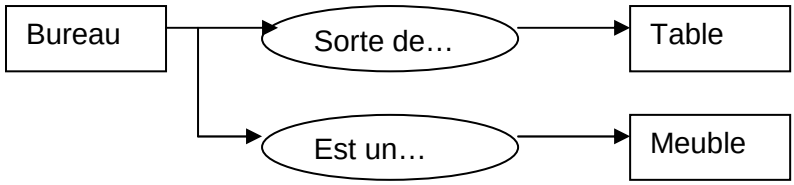


Figure 17 : Exemple de frame.

De par leur définition, les frames sont proches des objets de l'informatique. Toutefois, elles en diffèrent par le fait que les frames implémentent un langage de représentation, alors que nous voyons plutôt l'objet comme un langage de programmation, donc beaucoup plus rigide. Dans les systèmes de traitement d'images et de vision, les frames ont été largement utilisées.

C'est par exemple le cas dans le système VISIONS de Hanson ([Hanson77]), même si les frames ne sont pas explicitement nommées. Ce système adopte le découpage en trois niveaux d'abstraction :

- le bas niveau, qui considère les informations de type pixel,
- le niveau intermédiaire, qui prend en compte les primitives symboliques extraites à partir de l'image,
- le haut niveau qui traite de l'information sémantique.

A ce dernier niveau, un réseau est utilisé pour représenter la connaissance concernant les objets de la scène. A chaque nœud de ce réseau se trouve une structure nommée schéma.

Un schéma, (qui peut être vu comme une frame), est une structure contenant toute la connaissance nécessaire à la reconnaissance d'un objet. Il peut aussi représenter un concept de haut niveau, comme une scène, un objet sémantique ou une sous partie d'un objet. Dans les schémas, deux catégories de connaissances sont représentées : les connaissances descriptives et les connaissances stratégiques.

Les premières décrivent l'apparence des objets en prenant en compte le contexte (capteurs, échelles...).

Les secondes concernent les stratégies impliquées dans la reconnaissance des objets, la vérification de leurs relations spatiales et la résolution des conflits avec d'autres instances de schémas.

Le système SPAM de McKeown ([McKeown85]) exploite également le formalisme de règles. Celui-ci avait initialement pour but l'interprétation de scènes d'aéroports, mais a été étendu pour l'interprétation de zones suburbaines grâce à la mise en oeuvre d'un module d'acquisition de connaissances et d'un module d'évaluation de l'interprétation. La modélisation des connaissances a fait l'objet d'un gros effort.

Les objets sémantiques sont d'abord représentés par le formalisme des frames, qui sont ensuite transformées, par l'utilisation d'un module nommé RULEGEN, en un ensemble de règles.

L'acquisition de cette connaissance sous forme de frames peut se faire de deux façons, soit par un éditeur de frames en ligne, soit par l'exemple.

Une centaine de frames est utilisée, générant chacune 5 règles. Le module d'acquisition (ISCAN) permet une interaction entre le système et l'utilisateur. Ce dernier peut ainsi définir les objets du domaine considéré.

Les frames peuvent donc être utilisées pour la représentation des connaissances descriptives, opératoires et stratégiques. Elles permettent même souvent de représenter l'ensemble de ces connaissances dans un formalisme unifié. Toutefois, un tel "mélange" des connaissances tend à masquer le comportement du système et à le rendre relativement "incompréhensible".

1.2.4.3. Les systèmes à base de graphes.

Les graphes sont des réseaux sémantiques constituant un autre formalisme pour la représentation des connaissances. Les graphes sont composés d'un ensemble de nœuds et d'arcs.

Les nœuds comme les arcs peuvent être associés à des relations appelées attributs. Dans ce cas, les graphes sont alors appelés : "graphes relationnels attribués" (Figure 18). Ces graphes constituent un formalisme assez naturel pour la représentation des connaissances, les frame peuvent être considérés comme un type de graphe.

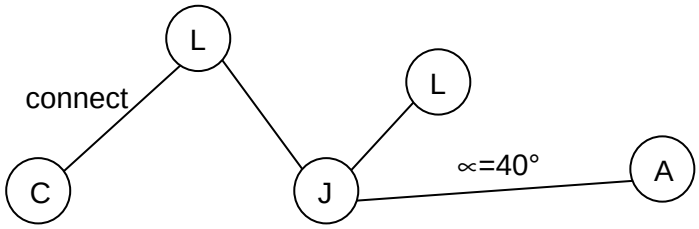


Figure 18 : graphe relationnel attribué, avec attibut : lien et angle

L'un des premiers systèmes à utiliser ce formalisme est le système ERNEST de Niemann ([Niemann90]). Ce système a pour vocation d'interpréter à la fois parole et image. Des applications concernant ces deux types de problèmes ont été montrées (reconnaissance de l'Allemand pour la parole et de scènes industrielles pour les images). La base du modèle dans ce système est un graphe modifié. Les nœuds y sont des structures de données complexes.

Trois types de nœuds sont définis :

- les concepts,
- les concepts modifiés (incomplets),
- les instances.

Cinq types d'arcs différents sont intégrés :

- les classiques « partie de », « spécialisation de » et « instance de »,
- mais aussi « concrétisation de » et « modèle de ».

Ces deux derniers arcs sont moins courants que les autres, ils expriment les relations entre différents niveaux d'abstraction des concepts.

En terme de stratégie, l'utilisateur choisit un nœud dans le réseau et celui-ci devient alors le but à atteindre. Les nœuds voisins sont alors recherchés, que ce soit au même niveau ou aux niveaux inférieurs, pour finir un algorithme permet d'ordonnancer les buts à atteindre. Dans ce système aucun formalisme de description de l'apparence graphique des objets n'est intégré.

Un deuxième système utilisant les graphes, comme formalisme pour la représentation des connaissances, est le système AIDA par Tönjes et Liedtke ([Tonjes99]), développé à l'université de Hanovre. Dans ce graphe, les nœuds représentent les objets de la scène. Les arcs du graphe sont de deux sortes :

- les arcs hiérarchiques,
- les arcs topologiques.

Les arcs hiérarchiques incluent les arcs de spécialisation ("sorte-de"), de constitution ("partie-de"), de concrétisation ("concrétisé-par") et des liens dits ("données de").

Ce dernier type d'arc constitue la grande différence avec le réseau d'ERNEST. Il permet en effet de faire le lien entre les concepts initiaux et leur représentation graphique en terme de primitives. Les arcs de type "données de" font l'interface entre l'aspect symbolique du graphe et l'image numérique.

Les arcs de type topologique permettent quant à eux de définir les relations spatiales entre les objets, transversalement aux couches formées par les arcs hiérarchiques.

Les graphes attribués permettent de représenter à la fois l'information structurelle et sémantique contenue dans des objets structurels. La topologie du graphe représente la structure globale de l'objet, alors que l'information sémantique attachée aux noeuds et arcs du graphe définie les propriétés locales. Un tel graphe peut être utilisé pour décrire la structure d'un certain objet en terme de ses composantes et des relations mutuelles qui existent entre elles.

Les approches à base de graphes semblent, grâce à leur aspect graphique, particulièrement adaptées pour une représentation des connaissances descriptives des systèmes. Toutefois, dans les systèmes évoqués, les moteurs d'interprétation associés à cette connaissance explicitée semblent monolithiques et le contrôle est alors problématique.

1.2.4.4. Les systèmes à base d'un formalisme orienté : données.

Les formalismes orientés sur les données ([Barry01]) sont des formalismes bas niveaux. Ils représentent la façon dont les données sont gérées. Ces formalismes sont : les piles, les files, les matrices, etc.

1.2.5. Les représentations.

Une représentation sert à "montrer" l'ontologie (cf § 1.2.3.2) au niveau symbolique, et à déclarer le formalisme (cf § 1.2.4) adopté par le système. Les représentations se déclinent en deux types :

- Les formats [Wilkinson00],
- Les langages [Kayser97].

Pour le domaine qui nous intéresse : en analyse de graphiques, les langages sont très peu utilisés.

Par contre les formats ont une place importante. Les formats permettant la gestion de l'image (au sens général du terme "image") sont divisés en deux types qui sont : les formats à base de "points" et les formats utilisant les "vecteurs". Pour finir nous verrons aussi les quelques nouveaux formats utilisant de la "sémantique".

1.2.5.1. Les formats images.

Ce sont historiquement les premiers types de format image. Ils devaient permettre une représentation de l'image sous forme matricielle.

Les formats à base de "points", stockent l'image comme une collection de Pixel[1]. Les trames de point sont représentées par des tableaux de valeurs. Les types les plus courants de formats de fichier sont :

- Le format "*Bitmap*" (BMP) est le plus courant et peut coder sur 24 bits les pixels. Mais ces fichiers sont volumineux pour une bonne qualité.
- Le format "*Tagged Image File Format*" (TIFF) est un format "propriétaire" d'Adobe. La quantification est de 24 bits, soit 16 millions de couleurs. Permet au choix de compresser ou non. Ce format existe en de très nombreuses variantes mais il fait actuellement office de standard pour la représentation des images dans les documents.

[1] Pixel de l'anglais : "PICture ELement".

- Le format "*Jpeg*[1] *File Interchange Format*" (JFIF) est une norme du *Joint Photographic Experts Group*. Ce format a une quantification sur 24 bits, soit 16 millions de couleurs. Il utilise une compression réglable avec perte par la transformée discrète en cosinus (acronyme anglais : DCT). Nouvelle version, le *JPEG2000* cf. 1.2.5.3

Ces types de format intègrent au mieux un algorithme de compression (avec ou sans perte) pour compenser la taille volumineuse des fichiers "bruts".

Mais le principal défaut des formats à "points" porte plus sur le type de contenu, dans le sens où les objets subissent des déformations avec les mises à l'échelle.

L'objectif des formats à base de vecteurs sera donc de résoudre si possible ces problèmes.

1.2.5.2. Les formats à base de vecteurs.

Dans l'ordre chronologique apparaissent alors les formats vectoriels. L'idée étant de répondre au redimensionnement sans pixellisation. Beaucoup des formats vectoriels se placent en tant que surcouches des formats images, ils permettent de représenter à la fois des données vectorielles et des données images.

Ce format code les images aussi bien sous formes géométriques que sous des formules mathématiques (droite, cercle, polygone, courbe de Bézier), ou encore d'image de point.

Les types communs de formats de fichier à base de vecteurs sont :

- Le format "*Portable Network Graphics*" (PNG) est une recommandation[2] du W3C[3]. La quantification est sur 8 ou 24 bits (vraies couleurs), le format inclut une compression sans perte des images non vectorielles.
- Le format "*Drawing Interchange Format*" (DXF) est un format propriétaire du logiciel de CAO[4] *AutoCAD*. Sauf exception, tous les logiciels de CAO savent gérer ce format.
- Le format "*Encapsulated PostScript*" (EPS) est largement utilisé dans les logiciels de bureautique.
- Le format "*Graphics Image File Format*" (GIFF) est un format "propriétaire" en deux versions : 87 et 89a. Il utilise un codage de 8 bits par Pixel soit 256 couleurs choisies dans une palette. Ce format permet une compression sans perte. Les fichiers GIF sont très répandus sur le web pour l'utilisation d'animations.
- Le format "*Scalable Vector Graphics*" (SVG) est recommandé par le W3C. SVG est une application du XML[5] dont l'objet est la description d'objets graphiques en 2D, sous forme vactoriel ou bitmap. Ce format supporte aussi l'animation.
- Le format "*ShockWave and Flash file format*" (SWF) est un format propriétaire de Macromédia. Ce format est axé sur l'animation vectorielle, mais permet néanmoins de travailler sur des images fixes. Les textes sont mal gérés à cause du concept de l'animation.

[1] JPEG : *Joint Photographic Experts Group*
[2] Datant d'octobre 1996.
[3] World Wide Web Consortium
[4] Conception Assistée par Ordinateur
[5] eXtented Markup Language

L'évolution chronologique va être de donner des informations aux éléments graphiques. Ces formats n'ont pas d'information sémantique entre objet.

Il faut toutefois noter que certains formats permettent l'utilisation de calques superposés, ce qui permet de situer en profondeur les éléments les uns par rapport aux autres, par exemple : "premier plan", "arrière plan".

1.2.5.3. Les formats incluant de la sémantique.

Il y a plusieurs types de sémantiques. Ainsi certaines "métadonnées" ont été liées par rapport au contexte de l'image, les formats sont de type "point" car les images sont souvent des photos. On trouve par exemple :

- Le format "*JPEG2000*" est un format à point (non vectoriel) qui permet une compresion multirésolution (par ondelettes[1]) dans un même fichier. Il utilise des métadonnées (auteur, date, etc.)
- Le format "*Exchangeable Image File Format*" (EXIF) donne des métadonnées techniques relatives à la prise de vue fournies automatiquement par l'appareil photo numérique (date et heure, ouverture du diaphragme, vitesse d'obturation, utilisation du flash, taille de l'image, etc.)
- Le format "*International Press and Telecommunications Council*" (IPTC) inclus des métadonnées ayant trait à la sémantique de l'image (légende, mots-clés, crédits, date, lieu, copyright)

D'autre part, même s'il ne s'agit pas à proprement parler de format, on trouve des normes ISO[2] capables de représenter des données sémantiques :

- *Graphical Kernel System* (GKS), cette norme de 1978 permet la manipulation d'image de façon indépendante des périphériques matériels pour la portabilité des applications. GKS repose sur les concepts de primitives et d'attributs graphiques. GKS identifie et normalise le contenu d'une page et le décrit avec 4 primitives graphiques : traits, surfaces, textes, polylines; chaque primitive possède des attributs: épaisseur, contour ou remplissage, couleur, fonte, taille.
- *Programmer Hierarchical Interactive Graphics System* (PHIGS), qui est un ensemble de services pour créer, visualiser et interagir avec des objets géométriques en 3D. Ceci indépendamment du matériel et des systèmes d'exploitation. Cette norme permet une structuration hiérarchique des formes géométriques et utilise une grande partie des concepts: primitives, attributs.
- Le format *Computer Graphics Metafile* (CGM), que l'on retrouve couramment dans beaucoup de logiciel de traitement d'image (par exemple, Photoshop) grâce à sa pérennité. CGM permet des constructions d'image par des compositions des vecteurs et d'image de pixel. CGM repose sur des primitives graphiques, des métadonnées et de la hiérarchie et des liens entre les objets.

[1] en anglais DWT : discrete wavelet transform
[2] International Standards Organisation

1.2.6. Conclusion sur les formalismes et représentations.

Afin de faire le bilan sur les formalismes et les représentations des connaissances en analyse d'images de documents graphiques, quelques systèmes sont donnés dans le tableau suivant (Tableau 2)

	Formalismes	Représentations
SVG (2001)	Vectoriel + listes	Langage balisé XML
DXF (1980)	Vectoriel + listes	Format de données
CGM (1997)	Vectoriel + graphes+ symbolique[1]	Format de données
Ah Soon ([AhSoon01])	Vectoriel + règles+ symbolique	Langage
Allanic ([Allanic00])	Vectoriel + graphes	Format de données
Hilaire ([Hilaire01])	Vectoriel + graphes	…
Pasternak ([Pasternak95])	Vectoriel + règles + symbolique	Langage
Ramel ([Ramel00])	Vectoriel + graphes	…
Song ([Song02])	Vectoriel + graphes	…

Tableau 2 : récapitulatif des formalismes et représentation des connaissances graphiques

On voit au travers de ce tableau que le formalisme vectoriel combiné avec le formalisme graphe constituent un mode privilégié de formalisation des connaissances graphiques en analyse d'image de documents. En effet il semble intuitif de les utiliser pour la représentation des primitives graphiques.

[1] Symbolique = graphes + labels

Chapitre 2 : Un formalisme et son opérationnalisation pour les connaissances graphiques.

Chapitre 2. Un formalisme et son opérationnalisation pour les connaissances graphiques.

2.1. Le processus de modélisation

Nathalie Aussenac-Gilles propose (cf. [Aussenac92]) un cycle de construction d'un système à base de connaissances qui consiste dans un premier temps à recueillir des connaissances préliminaires, à construire le schéma du modèle conceptuel du système, puis à acquérir les connaissances pour instancier ce modèle. Dans un second temps, le modèle complet du système obtenu doit être opérationnalisé et rendu exécutable sur un support informatique.

2.1.1. La construction du modèle.

Plusieurs analyses décrivent la construction du modèle d'une application sous la forme d'un processus qui comporte quatre étapes voir la Figure 19 et sert plus ou moins de référence (voir [Aussenac92])

Les quatre étapes se succèdent comme suit :

1. le recueil de données brutes,
2. la construction d'un schéma de modèle conceptuel,
3. la définition du modèle conceptuel complet,
4. l'implémentation de ce dernier dans une base opérationnelle de connaissances.

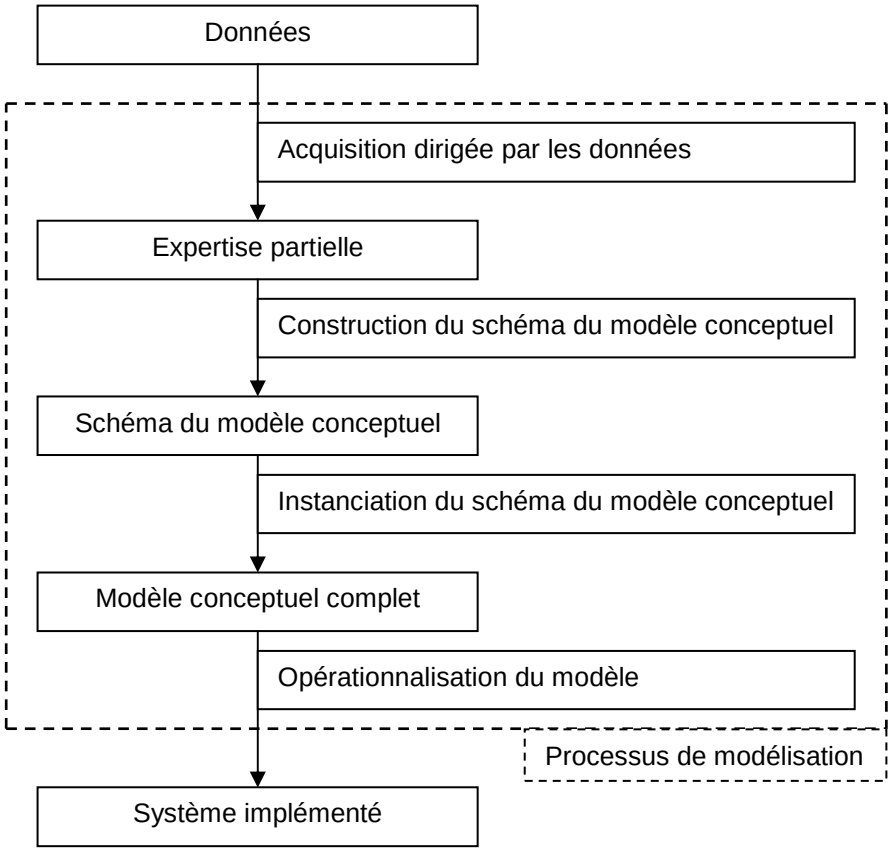


Figure 19 : La construction du modèle.

L'étape 1 est le fruit d'études en psychologie, ergonomie et IA. Les différents travaux de recherches menés dans ces domaines ont aboutit à des propositions de méthodes informelles comme des interviews et d'autres plus formelles comme les grilles répertoires[1]. Toutes correspondent à des démarches dites ascendantes[2]. Plus récemment, des outils d'analyse de corpus permettent de proposer des méthodes performantes, en particulier pour recueillir des données brutes en vue de l'élaboration de connaissances ontologiques.

Les étapes 2 et 3 concourent à l'élaboration du modèle conceptuel. En effet l'étape 2 a pour tâche de déterminer un schéma de ce modèle, c'est-à-dire un schéma des connaissances du domaine et un schéma des connaissances de raisonnement.

L'étape 3 consiste en un enrichissement des schémas de l'étape 2, avec les données du domaine d'application. Si ce découpage est intéressant, il n'est pas toujours exactement suivi dans la réalité dans la mesure où le schéma peut se construire durant l'élaboration du modèle conceptuel, en particulier dans les méthodes ascendantes.

L'étape 4 correspond au choix d'un langage pour l'implémentation du système final. Les différentes recherches dans ce domaine ont porté et portent sur la nature de ce langage et ses capacités inférentielles. Cette opérationnalisation est souvent l'occasion de réaliser des validations sur le comportement et l'utilisation des connaissances du système réel.

2.1.2. La méthodologie.

L'acquisition de la connaissance et la compréhension des méthodes de résolution à utiliser sont associées par le recueil et l'analyse, pour obtenir un modèle spécifique. Le modèle d'expertise est utile comme étape intermédiaire dans le développement de SBC.

Le modèle est indépendant des structures de représentation et des algorithmes dans le SBC. Il est conseillé d'utiliser un langage spécifique non-orienté vers l'implémentation, de type semi-formel complété seulement par la suite par un langage formel.

L'acquisition des connaissances (choix et analyse) peut être guidée par la réutilisation de modèles ou de composants génériques.

Lors de l'extraction des connaissances, il demeure quelques difficultés. En effet, il faut réussir à faire expliciter, capturer des connaissances explicites et les interpréter, cela sans introduire de biais.

Des travaux de recherche en psychologie, ergonomie et IA permettent d'établir des liens entre les types de tâches et les techniques d'extraction. Ainsi, il existe des techniques de recueil qui sont automatisées par des outils informatiques, (tels que : ETS, AQUINAS, PC_PACK en amont de CommonKADS) en utilisant des méthodes de type MACAO, ou KOD

Quelle que soit la technique utilisée pour modéliser la connaissance, le modèle est le fruit d'un travail humain. Le concepteur va donc malgré son sérieux étudier les connaissances avec sa sensibilité. De même une trop grande expérience impose parfois à tort une idée a priori du modèle. Il est donc conseillé de procéder à de nombreux tests sur des cas réels pour valider le côté générique du modèle.

[1] par Boose & Bradshaw (1987)
[2] terminologie employée par N.Aussenac-Gilles (1989)

2.1.3. Conclusion sur le processus de modélisation.

Les techniques à retenir afin de réaliser le modèle sont données par R Dieng (voir [Dieng98]) sous forme d'étapes :

- Détection des besoins,
- Construction du modèle,
- Diffusion du modèle,
- Utilisation du modèle,
- Evaluation du modèle,
- Entretien et évolution du modèle.

C'est à dire qu'après une phase d'observations, et d'analyse du travail (des simulations, des descriptions et explications), interviennent les reformulations, et les entretiens centrés. On peut compléter par des travaux à partir de corpus.

2.2. Le formalisme adopté.

Le paragraphe précédent (2.1) propose une méthodologie axée sur le recueil d'informations liées à la connaissance étudiée.

Cette étude a permis de gérer les informations permettant la connaissance, en effet la connaissance d'un point semble évidente par ces coordonnées. Mais qu'en est il pour un arc de cercle ? Une courbe mathématique ?

De plus notre modélisation doit apporter une connaissance sur les liens entre les objets, comment structurer les éléments entre eux pour indiquer les liens qui les unissent ?

Par une étude fondée sur les différents formalismes et formats présentés au §1.2.4, cette partie se propose d'indiquer les fondamentaux du formalisme adopté.

Le formalisme qui est proposé se décompose en deux niveaux :

- Un formalisme "orienté données" pour les descriptions des primitives graphiques et repose sur la notion d'objets.
- Un formalisme pour la structuration des primitives graphiques, basé sur les listes et les graphes.

2.2.1. Le formalisme orienté sur les données.

Cette partie du formalisme forme une couche de bas niveaux proche des données. Ce formalisme permet les descriptions vectorielles des primitives graphiques en se basant sur deux types d'objets :

- L'objet graphique,
- l'attribut graphique.

L'objet graphique est défini de façon générique, et peut être spécialisé en tout type d'objets graphiques comme l'illustre la Figure 20.

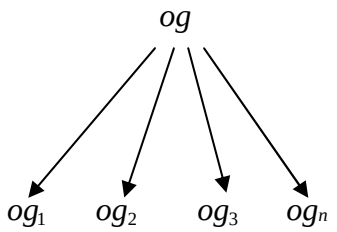


Figure 20 : L'objet graphique générique peut être spécialisé.

Comme tout objet, l'objet graphique est composé de données et de méthodes. Ainsi l'on peut noter un objet quelconque :

$$og_i = \{d, m\}$$

Les données membres peuvent être des données propres ou d'autres objets graphiques :

$$d = \{\{d_0, ..., d_j\}, \{og_1, ..., og_k\}\}$$

En parallèle il existe l'attribut graphique qui est utilisé en complément de l'objet graphique. Il permet de définir des relations entre objets graphiques.

L'attribut graphique générique peut être spécialisé comme le montre la Figure 21.

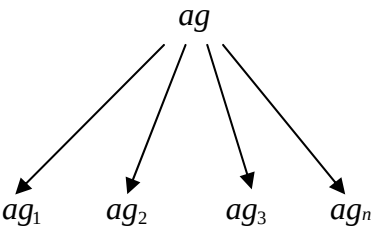


Figure 21 : L'attribut graphique générique peut être spécialisé.

Comme l'objet graphique, l'attribut graphique est composé de données et de méthodes. Ainsi l'on peut noter un attribut quelconque :

$$ag_i = \{d, m\}$$

Les données membres peuvent être des données propres ou d'autres attributs graphiques :

$$d = \{\{d_0, ..., d_j\}, \{ag_1, ..., ag_k\}\}$$

2.2.2. Le formalisme de structuration.

Implicitement dans cette étude, les objets graphiques sont décomposés en deux niveaux. Le niveau "orienté données" permettant la description des primitives graphiques ($p_1,\ldots p_n$) vient d'être décrite précédemment (2.2.1). Le formalisme pour la structuration de ces primitives graphiques (ls, g).

$$og \in \left(\{p_1,\ldots,p_n\},\{ls,g\}\right)$$

Ce formalisme de structuration repose sur les listes et les graphes

2.2.2.1. La structure liste.

La structure liste définit en deux ensemble : les objets graphiques : "*og*" et les attributs graphiques : "*ag*". La liste "*ls*" est notée :

$$ls=\{og,ag\}=\{\{og_1,\ldots,og_n\},\{ag_1,\ldots,ag_m\}\}$$

Les attributs sont définis de telle façon que pour un attribut donné d'indice : "*i*", il existe un couple d'objet graphique d'indices respectifs : "*i*" et "*j*", on obtient ainsi :

$$\forall ag_i \ \exists\{og_i,og_j\} \qquad f(og_i,og_j)=ag_i$$

2.2.2.2. La structure graphe.

2.2.2.2.1. Quelques rappels sur les graphes.

Ce paragraphe est un résumé de l'étude décrite dans [Nicolas02] ainsi que du travail mené dans [Lacomme03]. Un graphe est dit attribué si à chacun de ses noeuds et de ses arcs, est associée une valeur, un poids, une étiquette symbolique, ou/et un ensemble d'attributs numériques ou non (voir Figure 22.)

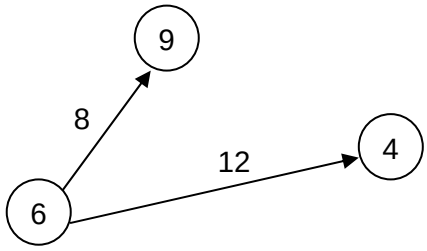


Figure 22 : exemple de graphe relationnel avec attributs de type "numérique"

Les définitions du Graphe Relationnel Attribué[1] que l'on peut trouver dans la littérature diffèrent quelque fois sensiblement entre elles, et il semble ne pas exister de formalisme unanimement adopté pour la définition de ce concept. Toutefois deux définitions relativement identiques et qui nous semblent intéressantes pour le formalisme présenté, sont données dans [Seong93] et [Messmer93]

[1] Attributed Relational Graph (ARG) en anglais

[Seong93] donne les définitions suivantes d'un ARG dans le cas où l'information attachée aux noeuds et aux arcs du graphe est une liste d'attributs quelconques.

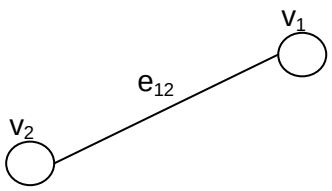
Définition 1 : [Seong93]

Un noeud attribué v et un arc attribué e sont associés à un ensemble d'attributs, dont chaque élément est une paire constituée d'un nom d'attribut et de sa valeur correspondant X . La valeur X de l'attribut peut être une valeur nominale, c'est à dire symbolique (soit une chaîne de caractères) ou une valeur numérique.

Définition 2 : [Seong93]

Un graphe attribué est un graphe $G_a = (V_a, E_a)$ où $V_a = \{v_1, \ldots, v_p, \ldots, v_q, \ldots, v_n\}$ est un ensemble de nœuds attibués et $E_a = \{\ldots, e_{p,q}, \ldots\}$ est un ensemble d'arcs attribués. L'arc $e_{p,q}$ relie les nœuds v_p et v_q par une relation attribuée.

La Figure 23 fournit un exemple d'ARG tel qu'il est défini par [Seong 93] d'après les définitions précédentes :



$G_a = (V_a, E_a)$ $V_a = \{v_1, v_2\}$
$E_a = \{e_{12}\}$
v_1 = { (forme rectangle) (compacité 0,9) (aire 0,8) }
v_2 = { (forme triangle) (compacité 0,6) (aire 0,5) }
e_{12} = { (relation adjacence) (angle 0,9) }

Figure 23 : Exemple de graphe relationnel à attributs multiples.

D'autre part [Messmer93] donne une définition sensiblement identique.

Définition : [Messmer93]

Un graphe relationnel attribué est un 6-tuple G = (V, E, L, A, μ, v) où :

$V=\{v_1,v_2,...,v_n\}$ *est un ensemble de nœuds*

$E=\{e_1,e_2,...,e_n\}\subseteq V\times V$ *est un ensemble d'arcs*

L est un ensemble fini d'étiquettes symboliques

A est un ensemble fini de valeurs d'attributs

$\mu: V \mapsto L\times A^k$ *est une fonction d'étiquetage des nœuds*

$v: E \mapsto L\times A^k$ *est une fonction d'étiquetage des arcs*

Le sens des valeurs des attributs est déterminé par les étiquettes symboliques qui indiquent ce que représente l'objet ou la composante modélisée sous la forme d'un noeud, et la relation entre deux objets qui est modélisée sous la forme d'un arc reliant deux noeuds.

L'exemple suivant d'ARG donné dans [Messmer93] permet d'illustrer cette définition :

G = (V, E, L, A, μ, v)
V = {v_1, v_2}
E = {(v_1, v_2)}
L = {Rect, Circle}
A = R,k = 2
μ(v_1) = (Rect, 2, 3)
μ(v_2) = (Circle, 5, _)
v((v_1,v_2)) = (inside)

Dans cet exemple, les interactions physiques entre deux figures géométriques, à savoir un rectangle et un cercle, sont représentées sous forme d'un ARG. L'étiquette d'arc "*inside*" indique que le rectangle se trouve physiquement à l'intérieur du cercle. L'étiquette "*Rect*" permet d'identifier que le type d'objet considéré est un rectangle est donc que les deux attributs qui sont associés sont la longueur et la hauteur du rectangle. L'étiquette "*Circle*" indique elle que l'objet considéré est un cercle qui par une convention arbitrairement fixée possède deux attributs numériques. Le premier attribut représente le rayon du cercle qui vaut "*5*" dans cet exemple, alors que le second est ici un paramètre vide.

2.2.2.2.2. La structure graphe adoptée.

De manière équivalente aux listes (vues au § 2.2.2.1), le graphe définit deux ensembles : les nœuds : "*o*" et les arcs : "*e*" . Les nœuds représentent les objets graphiques

$$g=\{o,e\}=\{\{og_0,...,og_n\},\{e_0,...,e_m\}\}$$

Les arcs sont définis de telle façon que pour un arc donné, cet arc relie un couple d'objet d'indice : "i" et "j" par un attribut graphique, noté ainsi :

$$e=\{\{(i,j)\},ag\}$$

Le formalisme utilise une structuration à base de graphes relationnels pyramidaux [Jolion90].

2.2.2.3. Conclusion sur le formalisme de structuration.

On peut considérer la liste comme une sorte de graphe, cependant celle-ci est fréquemment utilisée pour la représentation de polyligne, liste de points, etc. La liste est une redondance du graphe mais permet une gestion optimisée car la liste favorise la factorisation d'objets graphiques dans les graphes.

2.2.3. Les objets et attributs standards.

Comme cela a été montré au § 2.2.1, le formalisme repose sur un objet graphique et un attribut graphique générique qui peut être spécialisé.

Les objets et attributs standards vont être présentés ici.

2.2.3.1. Les objets standards.

L'objet générique est spécialisé en objets standards :

$$og=\{pt,l,ac,q,cb,...\}$$

Les objets standards sont définis par des données propres et par des méthodes associées. Ainsi les objets standards sont :

- Le point : $pt=\{x,y,m\}$

Où "*x*" et "*y*" sont les données de l'objet **point** "*pt*", elles représentent les coordonnées du point dans le repère de référence. D'autre part "*m*" représente les méthodes associées au point (déplacement, distance euclidienne entre deux points, le point étudié est-il connexe ?, etc…)

- Le segment de droite[1] ou ligne: $l = \{pt_b, pt_e, m\}$

Où "pt_b" et "pt_e" sont les données de l'objet **ligne** "*l*", elles représentent les points de début et de fin du segment de droite. D'autre part "*m*" représente les méthodes associées à la ligne (déplacement, longueur, azimuth, etc…)

- L'arc de cercle : $ac = \{pt_b, pt_c, pt_e, t, m\}$

Où "pt_b", "pt_c", "pt_e", et "*t*" sont les données de l'objet **arc** "*ac*", elles représentent respectivement les points de début de l'arc, de centre du cercle, de fin de l'arc et le sens de parcours. D'autre part "*m*" représente les méthodes associées à l'arc (déplacement, longueur de l'arc (en radian), etc…)

- Le quadrilatère : $q = \{pt_1, pt_2, pt_3, pt_4, m\}$

Où "pt_1", "pt_2", "pt_3", et "pt_4" sont les données de l'objet **quadrilatère** "*q*", elles représentent les quatres points du quadrilatère. D'autre part "*m*" représente les méthodes associées à celui-ci (déplacement, etc…)

- La courbe "mathématique" : $cb = \{pt_1, pt_2, pt_3, pt_4, m\}$

Où "pt_1", "pt_2", "pt_3", et "pt_4" sont les données de l'objet **courbe** "*cb*", elles représentent les quatres points de contrôle d'une courbe de Bézier cubique (voir §3.4.2.3). D'autre part "*m*" représente les méthodes associées à celle-ci (déplacement, etc…)

2.2.3.2. Les attributs standards.

L'attribut générique est spécialisé en attributs standards :

$$ag = \{\alpha, d, lbl, li, \ldots\}$$

Les attributs standards sont :

- L'angle :

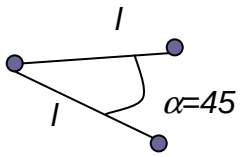


Figure 24 : Attribut "angle".

Il s'agit de l'angle entre deux objets, dans l'exemple de la Figure 24, les deux objets sont des lignes

- La distance :

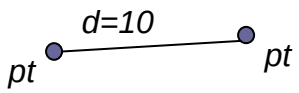


Figure 25 : Attribut "distance"

[1] Souvent remplacé par "*ligne*" traduction abusive de l'anglais "*line*"

Elle représente la distance euclidienne entre deux objets. Dans l'exemple de la Figure 25, la distance entre les deux points est de 10 pixels.

- L'étiquette (ou label) :

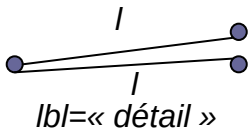


Figure 26 : Attribut "label"

Le label est une zone de texte alphanumérique, dans la Figure 26, la zone de texte contient le mot "*détail*". Cet étiquetage permet une gestion propre des attributs par un utilisateur dans un domaine quelconque.

- Le lien :

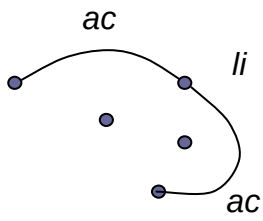


Figure 27 : Attribut "lien"

L'attribut "lien" permet de marquer deux objets liés entre eux comme les deux arcs de cercle donnés en exemple dans la Figure 27.

2.2.4. Mécanismes liés au formalisme.

Les connaissances dans un système sont internes ou externes à ce système. Ce paragraphe illustre comment les connaissances sont véhiculées dans notre approche entre les différentes parties du système d'analyse de graphiques, via les mécanismes :

- de représentation (ou externalisations) des connaissances graphiques dans une base
- de requête sur cette base

2.2.4.1. Mécanisme d'externalisation des connaissances.

Le mécanisme d'externalisation des connaissances permet d'exporter les connaissances vers l'extérieur ou de l'extérieur vers le système.

Chaque objet graphique est composé de données et de méthodes, il contient au travers de ces méthodes les mécanismes d'échange (Lecture/Ecriture) qui lui sont propres.

Par exemple un objet **point** va "*sortir*" ces données, comme illustré dans la Figure 28.

```
<OPoint x= "10" y= "10" />
```

Figure 28 : Exemple d'externalisation de l'objet : "point".

Si ce même objet **point** compose un objet **ligne** lors de la "*sortie*" des données de l'objet ligne (cf Figure 29), les coordonnées du **point** seront données par les méthodes de l'objet **point** et non pas de l'objet **ligne**.

```
<OLine length="14.1421" direction="0.785398">
        <OPoint x="10" y="10" />
        <OPoint x="20" y="20" />
</OLine>
```

Figure 29 : Exemple d'externalisation de l'objet : "ligne".

2.2.4.2. Mécanisme de requêtes.

L'échange des connaissances graphiques en interne du système repose sur un mécanisme de requêtes, élémentaire de type procédural, par le contenu (tests des types d'objet) et/ou par la structure (nombre d'objet, bouclage, etc.) appliqué aux graphes et/ou listes.

L'exemple schématisé à la Figure 30, illustre le propos :

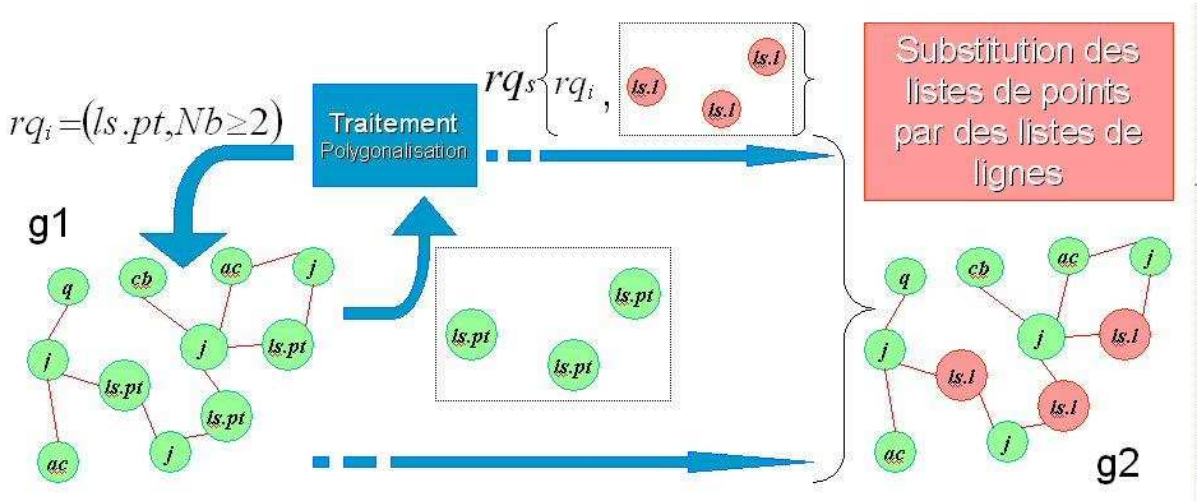


Figure 30 : Mécanisme de requête.

Soit un traitement ici une *polygonisation* (voir § 3.2) appliqué sur une représentation en graphe (*g1*). Le traitement va envoyer une requête initiale *rqi* sur ce graphe pour en extraire les liste de points (*ls.pt*) dont le nombre de points est supérieur à deux ($Nb \geq 2$).

Les listes de points sont extraites du graphe. Puis le traitement de *polygonisation* va remplacer dans le graphe initial (*g1*) les listes de points par des listes de lignes au travers d'une requête de sortie *rqs* qui utilise :

- la requête initiale *rqi* comme **trace** et
- les objets de remplacement les listes de lignes : *ls.l.*

On obtient alors un graphe *g2* ou les listes de points de *g1* ont été remplacées par des listes de lignes.

2.2.5. Exemple de l'échange des connaissances.

Cette conclusion repose sur l'exemple d'utilisation suivant : Soit l'image de la Figure 31. Cette image représente deux carrés liés par un segment de droite.

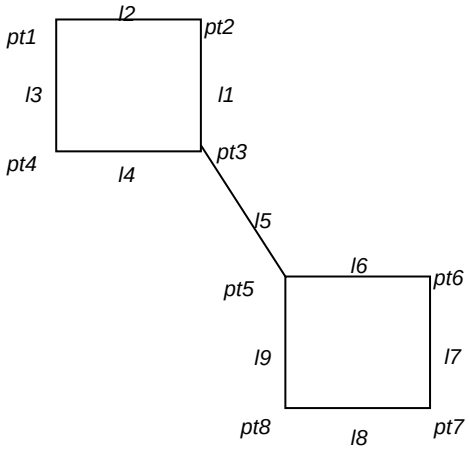


Figure 31 : Exemple du formalisme. Image de départ.

A partir de cette image, il est possible d'obtenir plusieurs représentations valides reposant sur la modélisation proposée.

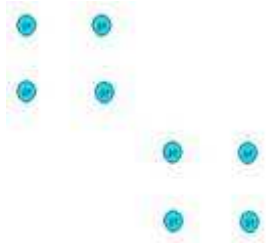


Figure 32 : liste de points.

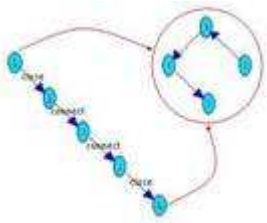


Figure 33 : liste hiérarchique.

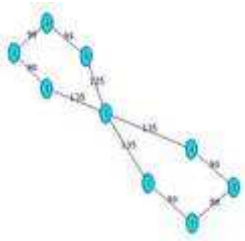


Figure 34 : graphe attribut "angle".

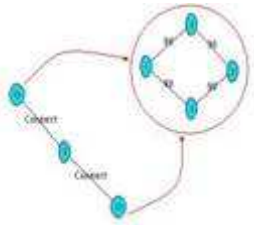


Figure 35 : graphe hiérarchique.

De la simple liste de points (Figure 32, ou lignes non illustrées ici) à la liste évoluée hiérarchique (Figure 31). Puis la modélisation en graphe attribué par un angle (Figure 34) pour finir par le graphe hiérarchique (Figure 33).

Toutes ces représentations sont possibles, aucune n'est privilégiée.

Chapitre 3 : Implémentation de quelques opérateurs pour l'extraction de primitives graphiques.

Chapitre 3. Implémentation de quelques opérateurs pour l'extraction de primitives graphiques.

3.1. Introduction.

Les opérateurs présentés dans ce chapitre ne sont pas utilisés directement sur l'image. Entre l'image et ces opérateurs d'extraction, d'autres algorithmes, sont utilisés. Ce sont ces algorithmes qui fournissent une structure de donnée adéquate aux opérateurs. Il s'agit des algorithmes de detection de contour, squeletisation ou chaînage.

Ces algorithmes de pré-traitements ne sont pas étudiés dans ce rapport.

Les opérateurs proposés sont les éléments de base pour l'extraction de primitives graphiques. En effet les trois groupes de traitements implémentés, permettent des approximations de types : segment de droites, formes bouclées, et courbes.

3.2. Polygonisation.

3.2.1. Méthode de la corde.

L'algorithme, proposé par V. Ramer [Ramer72] et repris par Douglas et Peucker [DouglasPeucker73], consiste à substituer la courbe par un ensemble de segments garantissant la distance minimale entre ceux-ci et la courbe. L'algorithme débute par les extrémités du trait à approximer. Puis, la distance orthogonale maximale entre la courbe et son approximation est utilisée comme contrainte (par rapport à un seuil) pour réaliser le découpage récursif de la courbe, voir Figure 36.

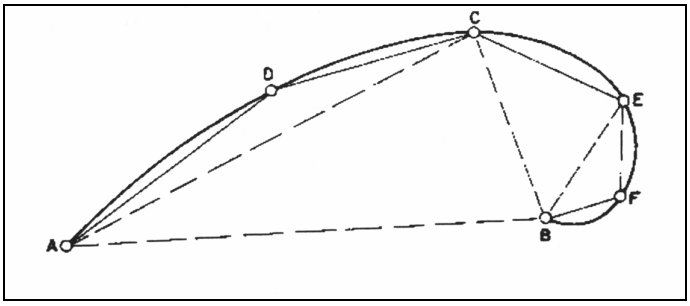


Figure 36 : Méthode de la corde par V. Ramer.

Cette méthode présente un handicap au bruit, mais permet une bonne détection des points anguleux.

L'autre handicap est la détermination du seuil, ce qui la rend peu généralisable. Il existe cependant des méthodes d'évaluation automatique de ce seuil (J. M. Ogier 1994.)

3.2.2. Méthode du Split & Merge.

Cette méthode est basée sur l'algorithme de la corde où la contrainte de distance orthogonale est utilisée pour diviser la courbe initiale en poly-lignes et pour fusionner des vecteurs consécutifs. Pavlidis et Horowitz [PavlidisHorowitz74] ont, en utilisant ce principe proposé la méthode du "*Split & Merge*" (diviser et fusionner).

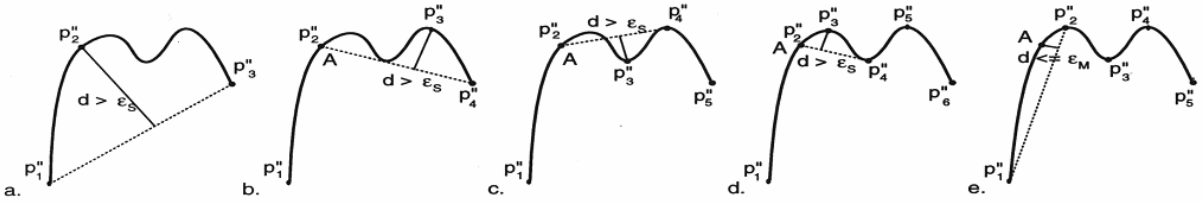


Figure 37 : méthode du Split & Merge.

La méthode du "*Split & Merge*" est aussi utilisée en vidéo, [Mirguet99] l'utilise dans un procédé de remplacement de motifs voici comment l'algorithme est défini :

Un critère ε est imposé, puis une corde est tirée entre le début et la fin de la courbe. Le point de la courbe, le plus éloigné de la corde, est détecté, voir Figure 37. Si la distance du point jusqu'au segment est supérieure à ε alors deux segments sont tracés. Un segment reliera le début de la courbe et le point, l'autre reliera le point et la fin de la courbe. Et ce de manière récursive jusqu'à ce que tous les segments se situent à une distance inférieure à ε de la courbe. Cette partie correspond au "*Split*" de l'algorithme.

La suite de l'algorithme consiste à diminuer le nombre de segments construits lors de la phase "*Split*" afin de simplifier l'approximation de la courbe. La procédure se déroule comme suit : on débute par les trois premiers sommets. La distance entre le deuxième sommet et la courbe est comparée avec le critère ε (le seuil.) Si la distance est inférieure, le premier et le troisième sommets sont reliés puis on reprend ces deux derniers points plus un nouveau et on compare à nouveau. Sinon, on prend trois sommets à partir du deuxième. Toute cette phase correspond à la fusion ou "*Merge*" de l'algorithme.

La méthode du *Split & Merge* permet de supprimer les sommets parasites ne correspondant pas à une courbure locale voir la Figure 38 et la Figure 39 ([Mirguet99]) et en particulier les courbures (roues, panneau de circulation, etc.) qui illustrent la méthode.

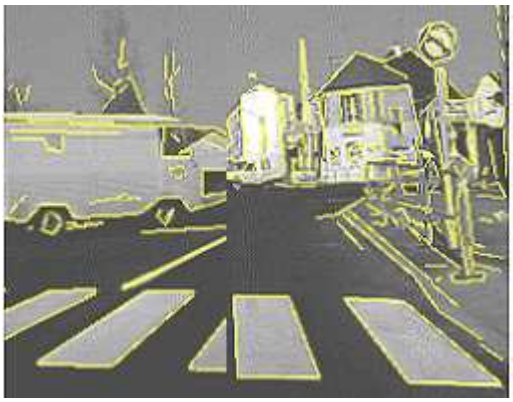


Figure 38 : Partie "*Split*" de l'algorithme.

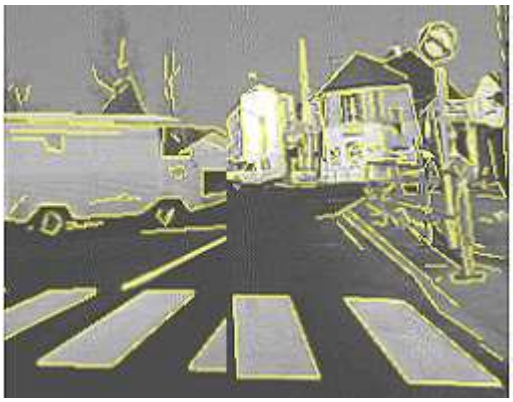


Figure 39 : Partie "*Merge*" de l'algorithme.

3.2.3. Méthodes basées sur l'aire entre courbe et approximation.

La seconde grande famille de méthodes de polygonisation est basée non plus sur des distances entre courbe initiale et approximation mais sur la surface située entre les deux. L'algorithme de base de cette méthode est celui défini par Wall et Danielson [WallDanielson84] qui consiste à remplacer chaque ligne par un ensemble de points de polygonisation en minimisant l'aire algébrique comprise entre la courbe et les segments approximés. Un nouveau point de polygonisation est construit dès que la surface entre la courbe et le segment de polygonisation atteint une taille fixée *a priori*. Dans certains articles, ce seuil prend en compte la longueur de l'arc initial, ce qui parait indispensable. Contrairement à la méthode de la corde décrite précédemment, cette méthode très globale permet de lisser les contours mais présente comme inconvénient une mauvaise détection des points anguleux qui se retrouvent fréquemment en dehors de la courbe initiale.

Même si les résultats paraissent inférieurs en termes de positionnement des points dominants, la notion d'aire est intéressante et probablement utilisable en complément de la méthode de la corde.

3.3. Appariement de contour.

L'appariement de contours se décompose en deux grandes phases :

- Détection de contours et vectorisation
- Appariement de vecteurs afin de construire des quadrilatères

La construction de quadrilatères dépend fortement de la qualité de la vectorisation des contours. En effet, des vecteurs de bonnes qualités permettent d'obtenir des quadrilatères non parasités car il y a théoriquement moins, de vecteurs non appariés.

Dans l'étude réalisée, la détection des contours et la vectorisation sont déjà réalisées. Les vecteurs utilisés, sont donc considérés comme de "bonnes qualités".

3.3.1. Appariement des vecteurs opposés[1].

Cette phase permet, à partir des vecteurs obtenus lors de la vectorisation, d'apparier les contours opposés selon des critères pour obtenir des quadrilatères.

Dans [Ramel00], J.Y. Ramel et al. décrivent une marche à suivre afin d'apparier les contours et construire ainsi des quadrilatères. La vectorisation donne une liste de vecteurs, qui est parcourue et testée pour chaque vecteur. Si le vecteur testé répond aux critères d'appariements avec un autre, alors un quadrilatère est construit, puis stocké. L'ensemble de la liste est ainsi parcouru jusqu'à ce qu'il n'y ait plus de vecteur à apparier.

Han et Fahn, dans [Han94], mettent plus en avant une idée de critère d'appariement qui se décomposerait en trois conditions :

- le parallélisme
- la superposition des vecteurs
- la distance entre les deux vecteurs

L'algorithme d'appariement de contours proposé ici se décompose en 5 étapes successives tout d'abord un **test d'appariement** utilisant trois critères logiques est effectué puis les critères sont **calculés**. Cette étape est suivie d'un **filtrage** des « appariements

[1] ou *Vector Matching* en anglais

éloignés ». Puis les **propositions** d'appariement subissent un tri logique, et pour finir l'**appariement** est réalisé.

3.3.1.1. Le test d'appariement.

Pour que les vecteurs puissent être appariés, il faut qu'ils répondent à trois conditions logiques qui sont :

- Les vecteurs ne doivent pas être connectés,
- Les vecteurs doivent être de sens opposés,
- Leurs projections sur l'axe des abscisses et des ordonnés devront être continues.

3.3.1.2. Le calcul des critères

Pour chaque proposition d'appariement, on détermine trois critères calculés qui nous aideront à déterminer les meilleures solutions. En effet, il se peut que les propositions trouvées premièrement ne soient pas très judicieuses, ou qu'elles soient répétées, ces critères nous permettent donc de sélectionner les solutions finales.

a. « gapStroke » : nous donne une pseudo représentation de la distance entre deux vecteurs voir Figure 40.

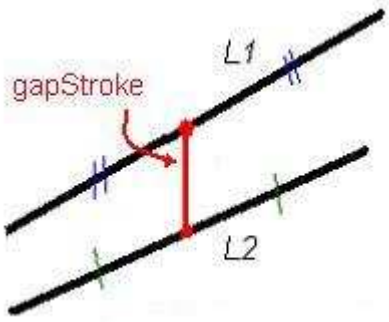


Figure 40 : gapStroke entre deux vecteurs.

Par la relation suivante :

$$gapStroke = \frac{longueur\ d}{\frac{(longueur\ L1 + longueur\ L2)}{2}}$$

b. « matchLength » : détermine si les deux vecteurs ont à peu près la même longueur

$$matchLength = \frac{longueur\ max - longueur\ min}{longueur\ min}$$

c. « matchTeta » : détermine si les deux vecteurs sont parallèles ou non

3.3.2. Structuration des quadrilatères construits

La génération des quadrilatères à plusieurs applications différentes. Dans la littérature, on voit apparaître des utilisations diverses. Mais toutes ces utilisations débutent par une même étape : la structuration.

Ainsi dans [Chang99] et [Tombre00], il ressort plusieurs utilités, telles que les relations de voisinages entre quadrilatères, la détection de jonction ou encore la correction, etc. …

Première étape : récupération des vecteurs non appariés

A la suite de la phase de création des quadrilatères, il y a forcément des vecteurs non appariés. Pour que la structuration soit bonne et qu'elle puisse déboucher sur une correction par exemple, il est important de récupérer ces vecteurs.

Nous rajoutons cette étape à la phase de Contour Matching :

. Marquage des vecteurs appariés à l'aide d'un tableau de booléen
. Copie des vecteurs non appariés dans une nouvelle liste
. Copie des vecteurs non appariés et des quadrilatères dans une autre liste

Deuxième étape : construction d'un graphe de relation

On peut désormais, ayant les quadrilatères et les vecteurs non appariés, construire un graphe de relation. On pourra ainsi, comme son nom l'indique, déterminer les relations entre objets (quadrilatère ou vecteur). Ceci, bien sûr, en vue d'une exploitation ultérieure telle que la détection de jonction ou la correction, voir les deux pourquoi pas ?

A partir de la liste de quadrilatères et de vecteurs non appariés, nous établissons les relations entre ces objets :

- Transformation, des objets en nœuds
- Détermination des relations entre nœuds:
  - . 2 quadrilatères connectés
  - . 2 vecteurs connectés
  - . 1 point commun entre 2 quadrilatères
  - . 1 quadrilatère connecté à 1 vecteur et création des arcs

3.3.3. Conclusion

C'est uniquement à la suite de cette construction de graphe que l'on peut commencer à parler de correction.

Cette correction aura pour but d'améliorer la qualité des quadrilatères, c'est-à-dire de diminuer au maximum le nombre de vecteurs non appariés.

Les perspectives seraient : le Split & Merge de quadrilatères, les corrections des extrémités, l'élimination des aberrations.

3.4. Approximation des courbes.

Les méthodes d'approximation des arcs n'ont pas été implémentées. Mais l'analogie entre les arcs et les courbes fait qu'il existe une complémentarité entre les méthodes d'approximation.

3.4.1. Les méthodes d'approximations d'arcs.

3.4.1.1. Méthodes de D. ELLIMAN.

D. Elliman, dans [Elliman01], s'inspire des travaux de Dori en ayant au préalable lissé le squelette dans la mesure où les extrémités des poly-lignes sont potentiellement des points de jonction que l'algorithme de vectorisation ne gère pas convenablement.

L'image binaire est vectorisée en utilisant l'algorithme d'Elliman, ce qui aboutit à une représentation comportant des chaînes de vecteurs très petites (quelques pixels de longueur.) Avant d'essayer d'ajuster des segments sur les arcs, les données sont lissées pour enlever les points qui n'indiquent pas une courbure cohérente, mais qui s'écartent trop par rapport à une ligne des moindres carrés.
Ce qui donne des vecteurs plus longs en moyenne, et rend la suite du traitement plus efficace.

L'étape suivante consiste à adapter un arc le plus grand possible à tous les points de la courbe. Tout d'abord on vérifie que la courbe est correctement représentée par une ligne droite. Si ce n'est pas le cas, un arc supposé est adapté. Ceci est fait par une corde entre l'extrémité de la courbe et le point d trouvé, qui est le plus éloigné de cette corde. Un arc est adapté par ce point et les deux points finaux. La distance maximum des points restants de la courbe à cet arc est alors calculée comme ceci est montré dans la Figure 41.

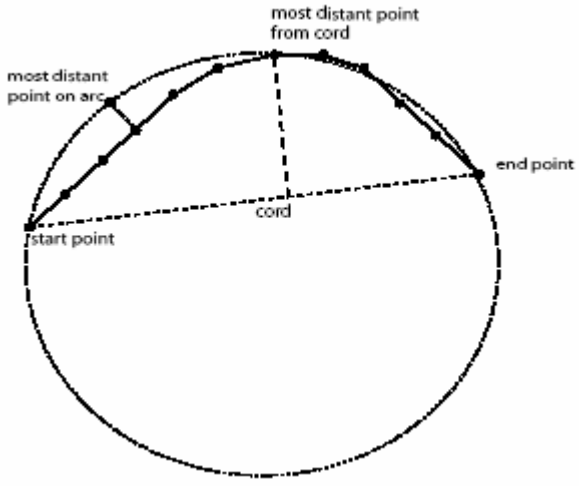


Figure 41 : La construction pour l'ajustage précis d'un arc supposé

Si la distance est inférieure à un certain seuil alors l'arc est accepté comme représentation de la courbe. Cependant, si cela échoue une tentative est faite pour adapter deux nouveaux arcs à n-1 points. L'un est trouvé en abandonnant le premier point, et l'autre en abandonnant le dernier. Trois arcs possibles peuvent être essayés par exclusion de deux points d'une extrémité de la courbe ou un point de chaque extrémité. Il y a quatre arcs possibles qui peuvent être essayés en abandonnant trois points, et cinq manières d'en exclure quatre et ainsi de suite.

L'algorithme se termine lorsqu'il n'y a plus que trois points à gauche dans la chaîne car cela garantit alors qu'un arc sera retourné, mais ce n'est pas une garantie de représentation plausible de la courbe.

3.4.1.2. Méthodes de X. HILAIRE.

Dans [Hilaire01], il est fait présentation d'une méthode générale de vectorization, appelée RANVEC (pour : *"Ran the arc segmentation Contest" & "Vectorization"*.) Puisque RANVEC manipule la vectorisation des arcs et des courbes par des segments de droite.

L'idée est de proposer un algorithme qui peut trouver des arcs longs, épais avec une bonne exactitude, et cela sans fragmentation. Puisque les images d'entrée sont censées être bruitées, une étape de pré-traitement est prévue, pour éliminer le bruit et pour reconstituer les formes en même temps. Voir illustration Figure 42.

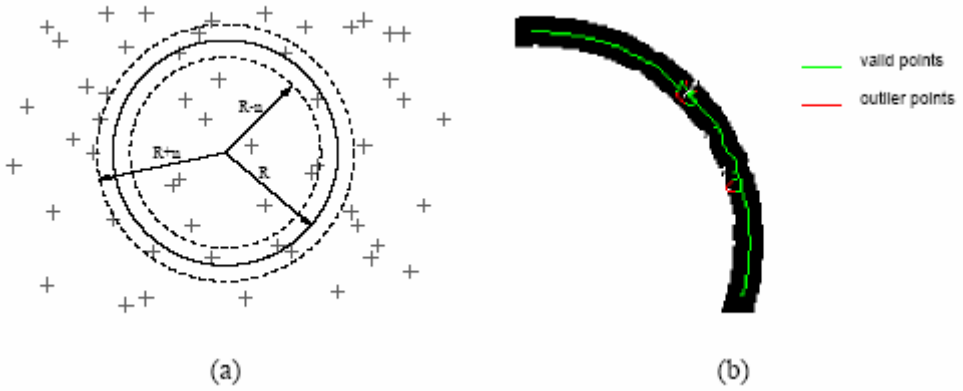


Figure 42 : (a) Caractérisation. (b) Effets du bruit aléatoire.

L'algorithme complet se compose de trois étapes :

1. Filtrage de l'image d'entrée,
2. Squelettisation de l'image filtrée, puis segmentation des chaînes et instanciation du squelette chaque vecteur en utilisant RANVEC,
3. Récupération des vecteurs réduits en fragments.

L'algorithme privilégie les traits épais et longs au détriment des arcs petits ou minces. En effet l'étape de filtrage donne en sortie uniquement les arcs importants,

Ainsi l'algorithme à partir d'un segment, propage des hypothèses sur les segments voisins puis valide chaque hypothèse par un critère de moindres carrés.

3.4.1.3. Méthode proposée par H. LOCTEAU.

Dans [Locteau04] la polygonalisation est exploitée par le biais de l'algorithme de la corde en imposant un seuillage automatique pour la distance du point le plus éloigné. La validation d'une hypothèse est effectuée via une distance de Hausdorff entre une composante et la représentation obtenue par l'algorithme de Bresenham.

L'algorithme débute par l'obtention d'un squelette simplifié. Puis un graphe est construit lors d'une phase de segmentation. Les nœuds du graphe sont des points de connexion et ses arcs des branches du squelette.

Les jonctions du squelette sont recherchées par l'intermédiaire de la segmentation du squelette. En effet cette phase isole les branches du squelette en composantes connexes. Les extrémités seules, de ces composantes, sont susceptibles d'avoir plus de deux voisins donc d'être des jonctions. Ainsi, un algorithme récursif construit une liste ordonnée de points formant les branches du squelette.

Puis intervient la phase de vectorisation des branches qui se découpe ainsi :

La première étape estime la distance entre une branche CC_c (ou sous-branche $CC_{c,s}$) et le segment S_c (respectivement $S_{c,s}$) reliant ses extrémités.

Puis une décision est prise sur la nature de l'objet détecté soit une ligne, soit un arc.

Enfin, la branche est coupée en deux, au niveau du point le plus éloigné de S_c lorsque la mesure de confiance des primitives est suffisante.

Les segments $S_{c,s}$ sont obtenus en utilisant l'algorithme de RAMER avec pour seuil : $||S_{c,s}|| \times \sqrt{3}/3$ soit $||MH|| \leq ||AB|| \times \sqrt{3}/3$ (notation de la Figure 43).

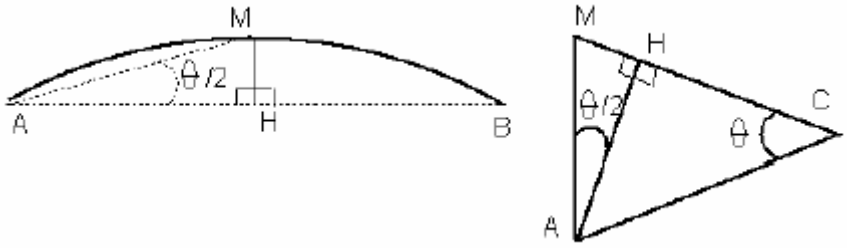


Figure 43 : Polygonalisation d'une branche du squelette

Il reste à estimer les paramètres de l'arc de cercle. Dans la Figure 43, une poly-ligne retenue par l'algorithme de Ramer y est représentée, A désigne le point ayant amorcé le segment et B la seconde extrémité fixée à travers le seuil (cf. paragraphe précédent).

La distance ||MH|| est le maximum entre $CC_{c,s}$ et S_c, cette distance induit le rapport ||MH||/||AH|| qui permet d'estimer l'angle $2 \times \theta$ correspondant à l'angle CA,CB où C désigne le centre de l'arc de cercle. Dans [Locteau04] ||AH|| est estimé comme étant égal à la moitié de ||AB||, bien que ce ne soit pas obligatoirement le cas. L'angle θ est donc obtenu par la formule suivante :

$$\theta = 2 \times \arctan\left(2 \times \frac{\|MH\|}{\|AB\|}\right)$$

Et le rayon par la formule :

$$\rho = \|AC\| = \|BC\| = \frac{1}{2} \times \frac{\|AB\|}{\sin(\theta)}$$

La position de C est alors déterminée en fonction des paramètres estimés et des angles de début et de fin de l'arc à vectoriser.

3.4.1.4. Transformée de Hough locale.

La transformée de Hough peut être utilisée pour détecter la présence de cercles dans une image [Bargeton03] mais nécessite au préalable l'utilisation d'une détection de contours.

La transformée de Hough trace des droites perpendiculaires aux contours des objets. Dans le cas d'un cercle toutes les droites vont converger vers le centre du dit-cercle comme dans la Figure 44.

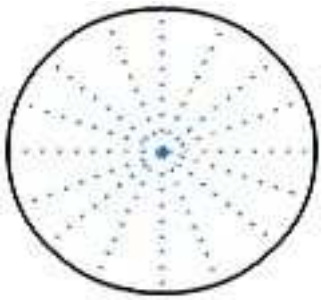


Figure 44 : Transformée de Hough d'un cercle.

Chaque point concentrant un maximum de droites perpendiculaires est considéré comme le centre d'un cercle. Enfin, il ne reste plus qu'à rechercher les cercles en utilisant chaque point comme un centre potentiel à partir duquel est tracé un cercle de rayon R. L'image de ce cercle est comparée à l'image des contours. Si elles sont identiques, il y a détection de cercle, sinon le rayon est incrémenté.

La transformée de Hough est longue en calcul, et les éléments de l'image autres que des cercles peuvent perturber l'analyse. C'est ce qui justifie l'intérêt de l'utilisation de cette méthode de façon localisée sur l'image.

3.4.2. Les méthodes d'approximations de courbes.

3.4.2.1. Méthodes de P.L. Rosin et G. West.

Cette méthode [Rosin95] est basée sur une approximation polygonale de D. Lowe (1987) qui est elle-même issue de la méthode de la corde.

Elle se décompose en deux étapes :

La première étape consiste en un découpage récursif de plus en plus fin de la courbe afin d'obtenir un arbre dont la racine est le segment de départ (celui qui relie les deux extrémités de la courbe), et chaque nœud est un segment intermédiaire sauf s'il s'agit d'une feuille.

La deuxième étape, vise à simplifier l'arbre obtenu.

La Figure 45 montre une courbe, et son approximation par segment, ainsi que les points de déviation maximum.

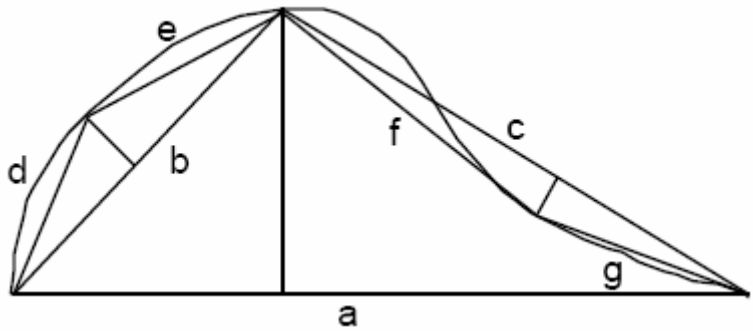


Figure 45 : Segmentation d'une courbe.

La Figure 46 montre l'arbre correspondant à la segmentation. Les nœuds cerclés sont considérés comme les plus significatifs dans l'arbre. C'est-à-dire, que "b" est plus

significatif que "d" et "e", comme "f" ou "g" sont plus significatifs que "c", et "b", "f" ou "g" sont plus significatif que "a". La représentation finale est bfg.

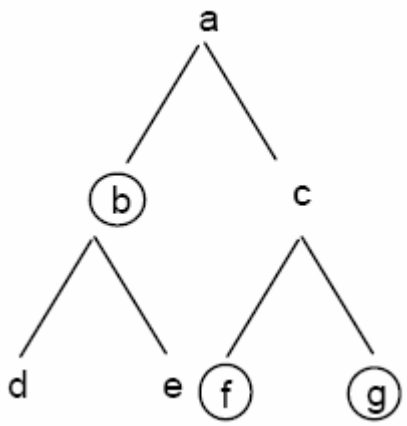


Figure 46 : Arbre correspondant.

3.4.2.2. Méthodes de P. L. Rosin.

Cette méthode [Rosin96] est basée sur l'approximation des cercles par des polygones à M cotés, (voir Figure 47) à l'aide d'un critère surfacique. On constate que les polygones sont réguliers de M=3 à 7, puis pour M>7, ils ne le sont plus, cela est dû à l'utilisation du critère surfacique (et non un critère de distance.). En effet l'algorithme crée des angles rentrant pour ajuster la surface.

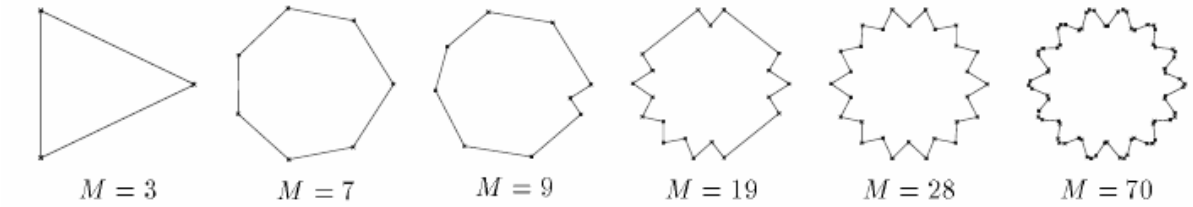


Figure 47 : Approximations d'un cercle par un polygone à M côtés.

L'algorithme donne pour M très élevé (exemple : M=70) une forme circulaire modulée par une sinusoïde voir Figure 48.

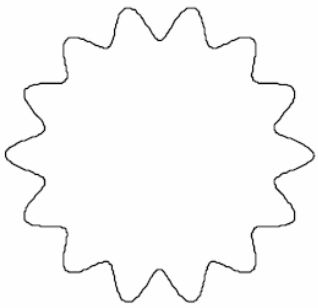


Figure 48 : Cercle modulé par une sinusoïde.

3.4.2.3. Méthodes d'interpolation par des courbes de Bézier cubiques.

Une courbe de Bézier est une fonction paramétrique, elle est caractérisée par : le nombre "n" de points de contrôle utilisés (son degré = n - 1), les coordonnées des points de contrôle : $P_i = \{P_0, P_1, \ldots, P_{n-1}\}$. De plus la courbe passe par les points P_0 et P_{n-1}.

L'expression de la courbe en fonction du paramètre "*t*" est :

$$P(t)=\sum_{i=0}^{n} P_i \times B_{i,n}(t)$$

Où les $B_{i,n}$ sont les polynômes de Bernstein :

$$B_{i,n}(t) = C_n^i \times t^i \times (1-t)^{n-i}$$

avec : $C_n^i = \frac{n!}{i! \times (n-i)!}$

Par exemple avec 4 points de contrôle (P0, P1, P2, P3 dans la Figure 49), on obtient une courbe de Bézier cubique, d'expression :

$$P(t) = (1-t)^3.P_0+3.t.(1-t)^2.P_1+3.t^2.(1-t).P_2+t^3.P_3$$

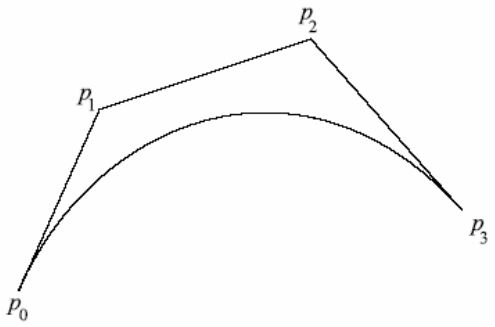


Figure 49 : courbe de Bézier cubique.

Une courbe de degré élevé peut se réduire à un ensemble de courbes cubiques mises bout à bout. Ainsi n'importe quelle courbe peut être représentée par un ensemble de courbes de Bézier cubiques ([Zorin02]).

Chapitre 4 : Mise en œuvre de l'interopérabilité entre opérateurs d'extraction.

Chapitre 4. Mise en œuvre de l'interopérabilité entre opérateurs d'extraction.

4.1. Mise en œuvre basée sur l'approche squelette.

Le premier exemple de mise en œuvre repose sur la chaîne de traitements illustrée à la Figure 50 :

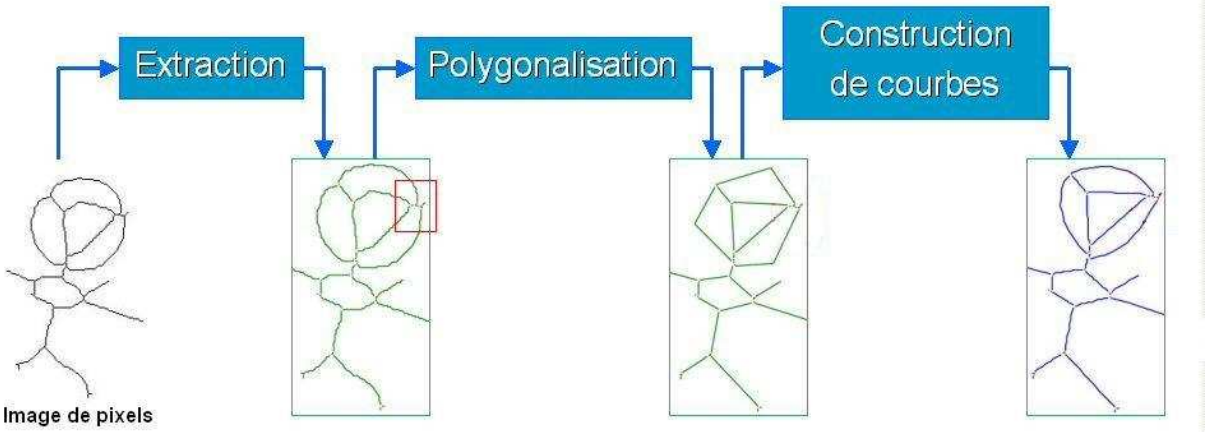


Figure 50 : Chaîne possible de traitements. Tout d'abord l'image originale représentant un squelette constitué uniquement de pixels va subir une **extraction** de ses primitives graphiques bas niveaux (voir le gros plan de la Figure 51) que sont les listes de points (en vert), les points isolés et les jonctions potentielles (en rouge).

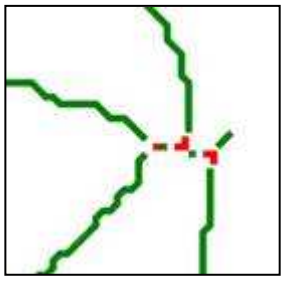


Figure 51 : Gros plan après extraction, listes de points en vert, jonctions en rouge.

Dans une seconde étape ces primitives graphiques bas niveaux vont être approximées par une polygonisation. La polygonisation va remplacer les listes de points (en vert sur la Figure 51) par des listes de lignes[1] (en vert sur la Figure 52)

[1] Il s'agit de segment de droite. (Abus dû à l'anglais "*line*")

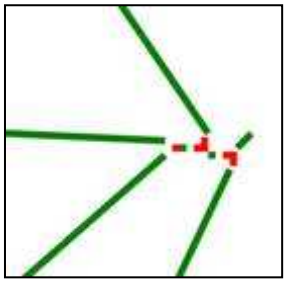


Figure 52 : Gros plan après polygonalisation, listes de lignes en vert, jonctions en rouge.

La dernière étape consiste en une interpolation des extrémités des segments (en vert sur la Figure 52) issus de la polygonisation par des courbes de Bézier cubiques (en bleu sur la Figure 53).

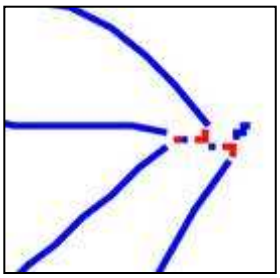


Figure 53 : Gros plan après Bézier, listes de courbes en bleu, jonctions en rouge.

Il est important de remarquer la non-destruction des jonctions potentielles (en rouge) au travers des différents traitements. En effet ces informations n'ont pas été détruites par ces différents traitements qui n'agissent pas dessus.

4.2. Mise en œuvre basée sur l'approche contour.

Le deuxième exemple de mise en œuvre repose sur l'approche à base de contour illustrée à la Figure 54. L'image originale représentant un contour constitué uniquement de pixels. Il est important de noter les effets de bords (cercles verts sur la Figure 54) de l'image de départ.

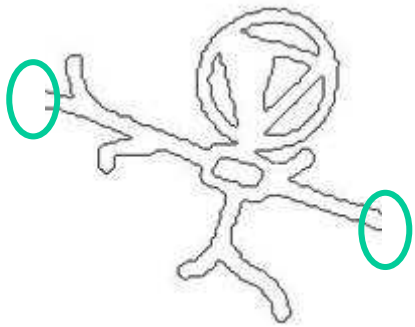


Figure 54 : Image originale (pixel) représentant le contour

Tout d'abord l'image originale va subir une **vectorisation** en vue d'obtenir ses primitives graphiques bas niveaux (voir la Figure 55) que sont les listes de lignes (en rouge),

et les listes de lignes bouclées (en vert). Les effets de bords ont permis la vectorisation de listes ouvertes sans avoir pour autant des occlusions.

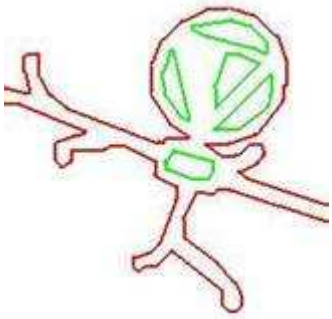


Figure 55 : Extraction des listes de lignes simples (en rouge) et bouclées (en vert)

L'étape finale consiste à former des quadrilatères. Pour se faire le traitement de **construction** utilise les listes de lignes bouclées (en vert sur la Figure 55) issues de la vectorisation pour les remplacer par des quadrilatères (en bleu sur la Figure 56).

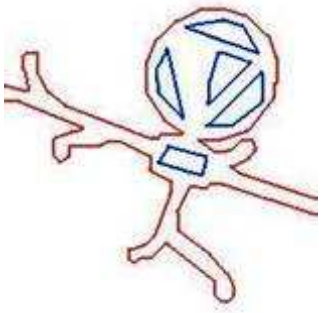


Figure 56 : Construction de quadrilatères en bleu, listes de lignes en rouge.

Ici comme dans le paragraphe précédent (4.2) les éléments non concernés par le traitement ne sont pas détruits. Ainsi les listes de lignes (en rouge) ne sont pas supprimées.

4.3. Conclusions et Perspectives.

4.3.1. Conclusions

Cette étude a énuméré différentes méthodes d'extraction de primitives graphiques. Puis la proposition d'une approche est présentée pour la gestion des connaissances graphiques extraites de l'image. Cette approche de gestion des connaissances, est basée sur un formalisme "générique" permettant plusieurs modélisations possibles pour une forme graphique donnée.

Le formalisme adopté, est basé sur des concepts orientés objets, en particulier pour la transmission, le polymorphisme et les propriétés d'extensibilité. Ce formalisme est disponible sous la forme d'une bibliothèque de modélisation.

La suite de cette étude se poursuit par la présentation des opérateurs d'extractions de primitives graphiques. Les traitements possibles sont :

- La corde
- Le Split & Merge
- Appariement de contour
- Interpolation par courbe de Bézier

Le sytèmes d'extraction fait appel à la bibliothèque de modélisation pour l'interopérabilité entre les opérateurs des connaissances graphiques, particulièrement pour leur combinaison.

4.3.2. Perspectives :

Les perspectives souhaitées, seraient à breve échéance : le développement d'une plateforme complète pour le traitement de reconnaissance graphique, basée sur le formalisme présenté.

A court terme, l'exploitation de l'interopérabilité entre les processings doit être validée pour développer quelques approches stratégiques.

Sur le long terme, le système devra être doté d'une interface évoluée pour permettre l'extraction des structures graphiques d'objet, à partir de la demande de graphique faite par l'utilisateur.

REFERENCES

[Ablameyko00] S. Ablameyko, T.P. Pridmore, “Machine Interpretation of Line Drawing Images”, Springer-Verlag, 2000.

[Adam98] S. Adam, Rapport de DEA ICSV et DESS AII, "Contribution au Développement d'un Système d'Interprétation de Plans de réseaux", 1998

[Adam01] S. Adam, PhD, "Interprétation de Documents Techniques : des Outils à leur Intégration dans un Système à Base de Connaissances", 2001.

[AhSoon98] C. Ah-Soon, PhD, "Analyse de Plans Architecturaux", 1998

[AhSoon01] C. Ah-Soon, K. Tombre, "Architectural Symbol Recognition Using a Network of Constraints", 2001

[Allanic00] H. Allanic, E. Petit, M. Villalon, F. Lopes, "Un Outil d'Interprétation d'Image Basé sur un Modèle Vectoriel Topologique", 2000

[Aussenac92] N. Aussenac-Gilles, J.-P. Krivine, J. Sallantin, Revue d'Intelligence Artificielle, "Introduction to the special issue on Knowledge Acquisition", 1992.

[Aussenac99] N. Aussenac-Gilles, J. Charlet, Cours, "Ingénierie des connaissances", 1999/2000, http://www.irit.fr/GRACQ/index-cours.html

[Bargeton03] A. Bargeton, "Détection des cercles dans une image (Hough Circle)", 2003, http://www.vieartificielle.com/article/index.php?action=article&id=188

[Barry01] C. Barry and C. Cormier and G. Kassel and J. Nobécourt, "Évaluation de Langages Opérationnels de Représentation d'Ontologies", 2001.

[Brooks83] R.A. Brooks, "Model-Based 3D Interpretations of 2D Images", IEEE Trans. Pattern Analysis and Machine Intellligence, vol. 5(2), pp. 140-150, 1983

[Chang99] F. Chang, Y.C. Lu, T. Pavlidis, "Feature analysis using line sweep thinning algorithm", 1999.

[Charlet98] J. Charlet, B. Bachimont, Assises GdR-PRC I^3 (Information Interaction Intelligence), 1998, http://sis.univ-tln.fr/gdri3

[Charlet02] J. Charlet, HdR, "L'Ingénierie des Connaissances Développements, Résultats et Perspectives pour la Gestion des Connaissances Médicales", 2002.

[Crevier97] D. Crevier and R. Lepage, "Knowledge-Based Image Understanding Systems: A Survey", 1997

[Delalandre03a] M. Delalandre, Y. Saidali, J.M. Ogier, E. Trupin, “Adaptable Vectorisation System Based on Strategic Knowledge and XML Representation Use”, Graphics Recognition (GREC), 2003.

[Delalandre03b] M. Delalandre, E. Trupin, J.M, Ogier, “Local Structural Analysis: A Primer”, Graphics Recognition (GREC), 2003.

[Dieng98] R. Dieng, O. Corby, A. Giboin, M. Ribière, "Methods and Tools for Corporate Knowledge Management", 1998.

[Dori99] D. Dori, "Syntactic and Semantic Graphics Recognition: The Role of the Object-Process Methodology", GREC 1999.

[DouglasPeucker73] D. H. Douglas, T. K. Peucker, "Algorithm for the reduction of the number of points required to represent a digitized line or its caricature". Can. Cartographer 10(2), 112-122, 1973.

[Elliman01] D. Elliman, "TIF2VEC, An Algorithm for Arc Segmentation in Engineering Drawings", 2001.

[Han94] C.C. Han, K.C. Fahn, "Skeleton generation of engineering drawings via contour matching", 1994.

[Hanson77] Allen R. Hanson, J. Prager, P. Nagin, R. Kohler, Edward M. Riseman: Segmentation Processes in the VISIONS System. IJCAI 1977, pp 642-643, 1977.

[Haton91] J.P Haton and al, " Le Raisonnement Humain en Intelligence Artificielle, Modèles, Techniques, et Architecture pour les Systèmes à Base de Connaissances", 1991.

[Hilaire01] X. Hilaire, "RANVEC and the Arc Segmentation Contest", 2001.

[Jolion90] J.M. Jolion, "Analyse d'images : Le modèle pyramidal, Traitement du signal", vol. 7, 5-17, 1990.

[Kayser97] D. Kayser, "La Représentation des Connaissances", 1997

[Lacomme03] P. Lacomme, Algorithmes de Graphes, 2003

[Lam95] L. Lam, C.Y. Suen, "An Evaluation of Parallel Thinning Algorithms for Character Recognition". Pattern Analysis and Machine Intelligence (PAMI), 1995.

[Lau02] K. K. Lau, P. C. Yuen, Y. Y. Tang, "Stroke Extraction and Stroke Sequence Estimation On Signatures", 2002.

[Levine78] M. Levine, "A knowledge-based computer vision system", Computer Vision Systems, A. Hanson and E. Riseman, Eds. New York: Academic, pp. 335-352, 1978.

[Lin02] F. Lin, X. Tang, "Off-line Handwritten Chinese Character Stroke Extraction", 2002.

[Locteau04] H. Locteau, S. Adam, E. Trupin, J. Labiche, P. Héroux, "Détection d'Arcs de Cercle par Comparaison du Tracé Théorique de Bresenham", 2004

[Lucas86] M. Lucas, "Algorithmes et Représentation des Données", 1986

[McKeown85] McKeown Jr., D. M., W. A. Harvey, J. McDermott,"Rule-Based Interpretation of Aerial Imagery", IEEE T-PAMI, vol 7(5), pp. 570-585, 1985.

[Messmer93] B.T Messmer, H. Bunke, "A network based approach for exact and inexact graph matching", Institut für Informatik und angewandte Mathematik, Université de Bern, Suisse, 1993.

[Minsky75] M. Minsky, "A framework for representing knowledge", The Psychology of Computer Vision, P. H. Winston, Ed. New York, McGraw-Hill, 1975.

[Mirguet99] L. Mirguet, "Remplacement de motifs dans une séquence animée", 1999 http://www.enseignement.polytechnique.fr/profs/informatique/Francois.Sillion/Majeure/Projets/mirguet/projet.html

[Nicolas02] S. Nicolas, "Vers une Bibliothèque pour la Manipulation et la Factorisation de Sous-Graphes", 2002.

[Niemann90] H. Niemann, G. F. Sagerer, S. Schröder, F. Kummert, "ERNEST : A Semantic Network System for Pattern Understanding", IEEE Trans. PAMI, vol. 12(9), pp. 883-905, 1990

[Pasternak95] B. Pasternak and B. Neumann, "The Role of Taxonomy in Drawing Interpretation", 1995
[PavlidisHorowitz74] T. Pavlidis, S. L. Horowitz, Segmentation of plane curve, IEEE Transactions on Computers, vol. C-23, 1974, 860-870.

[Ramel00] J.Y. Ramel, N. Vincent, H Emptoz, "A structural representation for understanding line-drawing images", 2000.

[Ramer72] V. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves", 1972.

[Rosin95] P. L. Rosin, Geoff A. W. West, "Nonparametric Segmentation of Curves into Various Representations", 1995.

[Rosin96] P. L. Rosin, "Techniques for Assessing Polygonal Approximations of Curves", 1996.

[Saidali02] Y. Saidali, PhD, "Modélisation et Acquisition de Connaissances : Application à une Plate-Forme de Traitement d'Images", 2002.

[Seong93] D. S. Seong, H. S. Kim, K. H. Park, "Incremental Clustering of Attributed Graphs", IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, No. 5, 1993.

[Song02] J. Song, F. Su, CL. Tai, S. Cai, "An Object-Oriented Progressive-Simplification based Vectorization System for Engineering Drawings: Model, Algorithm and Performance", 2002

[Song03] J. Song, M.R. Lyu, M. Cai, and S. Cai, "Graphic Object Recognition from Binary Images: a Survey and an Integrated Paradigm", 2003.

[Tombre98] K. Tombre, “Ten Years of Research in the Analysis of Graphics Documents, Achievements and Open Problems”, Image Processing and Image Understanding, 1998.

[Tombre99] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, S. Tabbone, "Stable and Robust Vectorization : How to Make the Right Choices", Graphics Recognition (GREC), 1999.

[Tombre00] K. Tombre, S. Tabbone, "Vectorization in graphics recognition : To thin or not to thin", 2000.

[Tonjes99] R. Tönjes, S. Growe, J. Bückner, C.-E. Liedtke, "Knowledge Based Interpretation of Remote Sensing Images Using Semantic Nets, Photogrammetric Engineering and Remote Sensing, vol. 65(7), pp. 811-821, 1999.

[Ullman89] J.D. Ullman, "Principles of Data-Base and Knowledge Base Systems", 1989

[VanDenElst96] J. Van Den Elst, PhD, "Modélisation de Connaissances pour le Pilotage de Programmes de Traitement d'Images", 1996.

[WallDanielson84] K. Wall, P.E. Danielson, "A fast sequential method for polygonal approximation of digitised curves", CVGIP 28, 1984.

[Wenyin99] L. Wenyin, D. Dori, "From Raster to Vectors : Extracting Visual Information from Line Drawings. Pattern Analysis and Applications (PAA), 2(2) : 10-21", 1999.

[Wilkinson00] L. Wilkinson, D.J. Rope, D.B. Carr, M.A. Rubin, "The Language of Graphics", 2000

[Zorin02] D. Zorin, "Bezier Curves and B-splines, Blossoming", 2002.