Institut National
Polytechnique de Lorraine

Centre de Recherche en
Informatique de Nancy

THESE DE DOCTORAT DE L'INPL

(Mention Informatique)

présentée à L'Institut National Polytechnique de Lorraine

par

Karl TOMBRE

La saisie automatisée de documents composites : reconnaissance, codage et interprétation des parties graphiques

soutenue le 15 juin 1987

Composition du jury :

Président :	J.P. HATON
Rapporteurs :	S. CASTAN C. PAIR
Examinateurs :	R. MOHR G. SIGNOR

Institut National
Polytechnique de Lorraine

Centre de Recherche en
Informatique de Nancy

THESE DE DOCTORAT DE L'INPL

(Mention Informatique)

présentée à L'Institut National Polytechnique de Lorraine

par

Karl TOMBRE

La saisie automatisée de documents composites : reconnaissance, codage et interprétation des parties graphiques

soutenue le 15 juin 1987

Composition du jury :

Président :	J.P. HATON
Rapporteurs :	S. CASTAN
	C. PAIR
Examinateurs :	R. MOHR
	G. SIGNOR

à la mémoire de mon grand-père Karl M. Tombre,
qui porta toujours un très vif intérêt aux
études que je faisais, et qui les facilita
aussi financièrement parlant.

Remerciements

Je tiens à remercier :

- Mr. Roger Mohr, professeur à l'Institut National Polytechnique de Lorraine, qui a encadré et dirigé mon travail de recherche. J'ai énormément bénéficié de sa compétence, de ses conseils et de ses encouragements. Ses nombreuses qualités scientifiques et humaines font de lui un chef avec lequel on travaille avec un réel plaisir, car il sait créer une ambiance propice à la fois à l'éclosion de nouvelles idées et au développement de relations amicales et détendues entre les membres d'une équipe. J'espère sincèrement que notre collaboration pourra continuer dans les années à venir.
- Mr. Jean-Paul Haton, professeur à l'Université de Nancy I, qui est l'instigateur du projet sur lequel porte cette thèse au sein de l'équipe Reconnaissance des Formes et Intelligence Artificielle du CRIN, et qui me fait l'honneur de présider ce jury. Je lui suis particulièrement reconnaissant pour le grand intérêt qu'il a toujours porté à mes travaux et pour les nombreux encouragements qu'il m'a prodigués au fil des ans. Ses conseils ont plusieurs fois été décisifs quand il s'agissait pour moi de choisir l'orientation de ma carrière scientifique.
- Mr. Claude Pair, professeur à l'Institut National Polytechnique de Lorraine, pour le grand honneur qu'il me fait d'avoir accepté de juger mon travail, malgré les nombreuses charges et responsabilités qu'il assume. Il a marqué de manière durable l'enseignement et la recherche en informatique à Nancy, et j'ai aussi pu tirer profit de sa rigueur scientifique au travers de ses remarques et conseils, qui m'ont permis d'apporter plusieurs améliorations à la qualité de cette thèse.
- Mr. Serge Castan, professeur à l'Université Paul Sabatier de Toulouse, qui a accepté d'être rapporteur de ma thèse et de siéger dans ce jury. Qu'il trouve ici l'expression de ma gratitude pour l'intérêt dont il fait ainsi preuve à l'égard de mon travail.
- Mr. Gilles Signor, responsable de projet à TITN, qui a rendu possible la collaboration entre TITN et le CRIN dans le cadre du projet HERODE. Cette collaboration m'a permis d'effectuer ce travail de recherche dans de très bonnes conditions financières. Je le remercie pour la confiance qu'il nous a ainsi témoignée et pour la bonne entente et l'esprit d'équipe qui se sont établis entre nos deux organismes.

Je veux aussi remercier tous mes collègues du CRIN pour l'atmosphère de travail sympathique dont j'ai bénéficié. Je tiens particulièrement à remercier ceux avec qui j'ai collaboré à un moment ou un autre. Merci à Gérald Masini pour sa très grande disponibilité et pour toutes ses remarques et critiques, aussi bien sur les programmes développés que sur les articles rédigés. Ses conseils ont toujours été très utiles, et m'ont souvent appris à être plus rigoureux. Je remercie aussi Brigitte Wrobel pour tous les moments qu'elle a consacrés à m'écouter ou à me relire. Je dois des remerciements chaleureux à Dominique Antoine, qui a beaucoup contribué à l'implantation des derniers modules de notre système, dans les derniers mois où j'étais tellement pris par la rédaction de cette thèse. Il m'est impossible de citer par nom tous mes amis du CRIN, la liste étant trop longue, mais je tiens à remercier tous ceux qui à un moment ou un autre ont pris de leur temps pour me parler, m'écouter ou me relire.

Un grand merci aussi aux collègues du projet HERODE, avec lesquels il a été agréable de travailler. Cette expérience de collaboration européenne a été très positive et intéressante pour moi. Je voudrais en particulier remercier Serge Seisen et Eric Meynieux de la société TITN ; nous avons formé une petite équipe de trois personnes qui a pu coopérer très étroitement malgré les distances. Je remercie également Wolfgang Postl de la société SIEMENS, qui a réalisé certaines parties de notre module.

Ma famille a joué et joue un rôle primordial dans ma vie. J'ai eu le privilège de grandir dans un foyer rempli d'amour et de chaleur, et cela donne un équilibre mental et une force intérieure qui sont souvent indispensables pour passer les caps difficiles. La sagesse et le chemin du bonheur valent infiniment plus que n'importe quel titre, et ce sont donc mes parents qui m'ont appris les choses les plus importantes de la vie. Ils m'ont aussi aidé à trouver le chemin de la vie éternelle, ce qui est encore plus grand. Je remercie également ma famille d'avoir supporté patiemment les horaires fantaisistes d'un chercheur en informatique.

Mes remerciements ne seraient pas complets s'ils n'incluaient pas ma grande famille spirituelle, tous mes amis, mes frères et soeurs, et tout particulièrement ceux de notre assemblée chrétienne de Lunéville. La communion que nous avons les uns avec les autres ne peut être décrite ou résumée en quelques lignes. Mais je suis convaincu que le fait d'avoir une telle communion, sur des bases n'ayant rien de commun avec mon métier, a été et continue à être pour moi un contrepoids indispensable pour que mes pensées ne soit pas «dévorées» par l'informatique... Je veux remercier en particulier Marc Auchet : menant lui-même une carrière universitaire, il m'a aidé par sa vie et son exemple à faire les bons choix, pour que le travail quotidien ne soit pas un obstacle au développement spirituel.

Chapitre 1

Introduction

1.1 La bureautique

Ces dernières années ont vu la naissance et le développement très rapide, dans les entreprises et les administrations, d'un nouvel outil appelé la **bureautique**. On la définit parfois comme l'ensemble des techniques et moyens électroniques concourant à l'automatisation des activités du bureau [50]. Les fonctions de la bureautique sont multiples : gestion d'agenda, de l'occupation des salles, systèmes d'aide à la décision tels que les tableurs, systèmes de suivi de dossiers. Toutefois, un de ses rôles principaux est la gestion automatisée de l'information présente sous forme de **documents**, au sens le plus général de ce terme.

C'est uniquement ce dernier aspect de la bureautique qui nous intéresse dans le cadre du travail présenté ici. La gestion de documents englobe des moyens de **production de documents**, de **communication** (messagerie, télécopie, transferts de documents électroniques, etc.) et de **stockage et archivage** des documents déjà produits.

Le **document électronique** est donc devenu un concept fondamental : le document est conçu, modifié, transmis et archivé sous forme électronique. On n'en sortira une copie sur papier que si le besoin s'en fait sentir ; le support principal restera électronique.

Du point de vue matériel, l'évolution tend vers la conception de **systèmes de bureautique intégrée**. Ce sont des stations de travail offrant dans un univers homogène les différents services que nous avons cités. Pour cela, elles ont besoin d'une puissance de calcul propre, pour les travaux de création et d'édition interactive, et d'accéder à un **réseau** pour les services de communication et pour que plusieurs postes de travail puissent partager des ressources communes. Ces ressources peuvent être des **imprimantes** alliant haute qualité et haut débit, comme les imprimantes laser ou électrostatiques de plus en plus répandues, des systèmes de **stockage** permettant l'archivage et le classement d'un grand nombre de documents, par le moyen de disques magnétiques de haute capacité ou de disques optiques numériques, ou même, comme nous allons le voir plus précisément dans ce qui suit, des outils de saisie et numérisation de documents papier, *scanners* ou caméras à haute résolution.

De plus, au niveau des stations de travail, l'accent est mis très fortement sur **l'interactivité** : grâce à des moyens matériels (souris, écran graphique, ...) et logiciels (système à menus déroulants, icones, ...), on fournit à l'utilisateur un environnement de travail convivial et une facilité d'emploi sans laquelle un non-informaticien ne se résoudrait jamais à exploiter complètement le potentiel du système dont il dispose.

1.2 La norme ODA / ODIF

A cause de ce développement rapide de la bureautique, le besoin s'est fait sentir de normaliser le concept de document électronique, pour que des systèmes hétérogènes puissent avoir une chance d'échanger des documents sous cette forme. L'effort de normalisation est mené par plusieurs organismes, dont en tout premier lieu l'organisation internationale de normalisation **ISO**, et l'association européenne des fabricants d'ordinateurs **ECMA**. Cette dernière a adopté en juin 1985 comme standard **ECMA 101**[1] la norme **ODA / ODIF**[2], qui spécifient l'architecture d'un document bureautique et le format sous lequel il peut être transféré vers un autre système. Horak [23] décrit de manière complète cette norme ; nous nous contenterons d'en présenter les grands traits.

ODA définit pour un document une double structure : la structure logique et la structure physique (*layout structure*). La première décrit le contenu du document comme une hiérarchie d'objets logiques, tels que *résumé*, *titre*, *chapitre*, *paragraphe*, *figure*, *table*, etc. La structure physique décrit une hiérarchie d'objets physiques, tels que des pages, des colonnes, des photos, etc. C'est bien sûr l'ordre logique du document qui est prépondérant. La structure logique est représentée par un arbre. C'est pour pouvoir présenter le document, sur papier par exemple, que la structure physique doit être déterminée ; les différents objets logiques composant le document donnant naissance, étant données les contraintes de présentation, à un certain nombre d'objets physiques.

Les objets ont aussi des **propriétés** comme leur dimension, et des relations avec d'autres objets. Ces relations sont de plusieurs types : relations hiérarchiques entre les objets composites et leurs constituants plus élémentaires, relations entre objets logiques, comme une référence à une figure dans un texte, relations entre objets physiques comme par exemple l'ordre dans lequel des blocs superposés doivent être affichés ou imprimés, et relations entre objets logiques et objets physiques, appelées souvent les directives de formattage, qui permettent à l'objet logique de garder un certain contrôle sur la manière dont il sera présenté.

Aux feuilles de cet arbre double, on trouve le **contenu** proprement dit, c'est-à-dire le texte, les données photographiques et les figures géométriques qui portent l'information destinée à l'être humain. Un document est donc décrit par sa structure logique, sa structure physique et son contenu. La figure 1 résume cette structuration.

---

1 ECMA 101 Standard ISO DP / 8613

2 ODA : Office Document Architecture, ODIF : Office Document Interchange Format

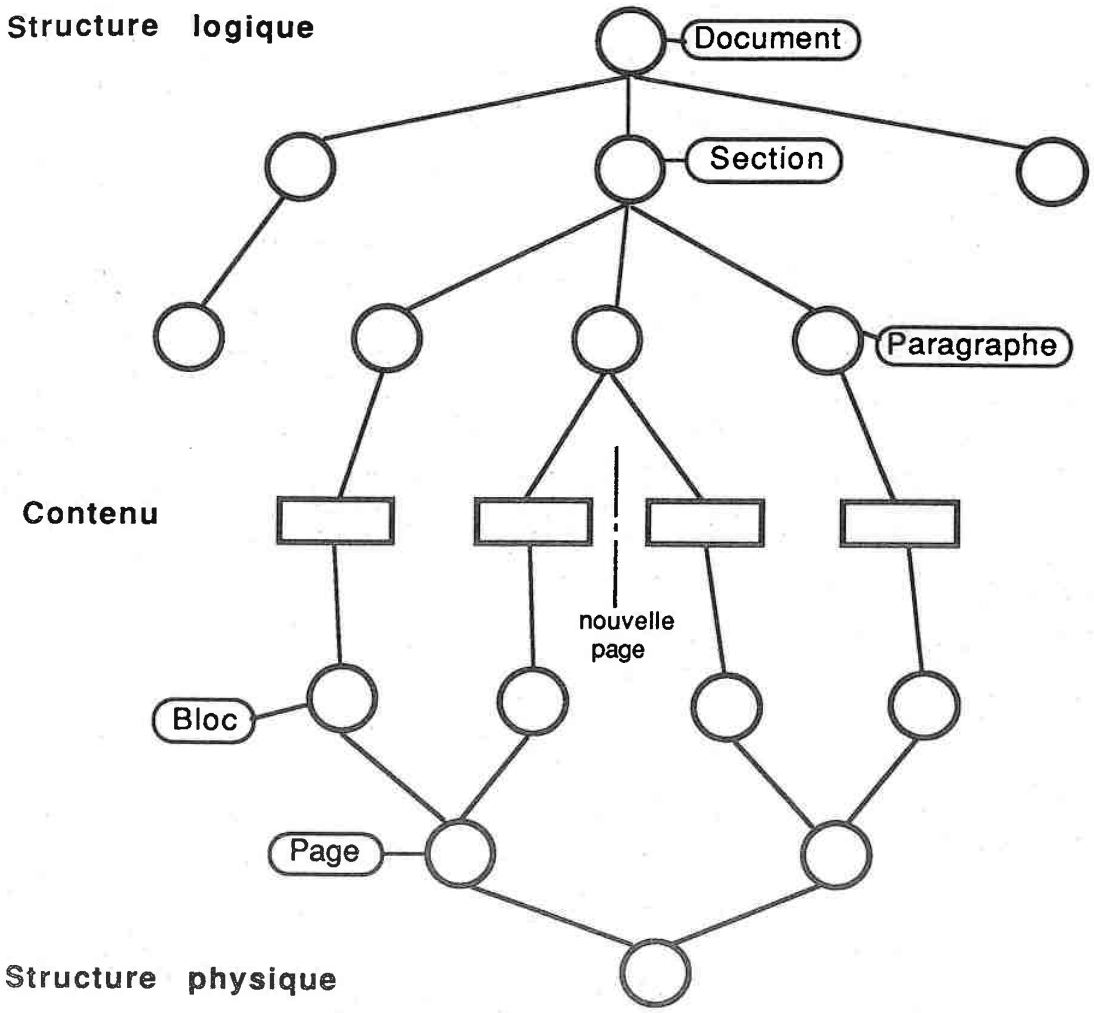


Figure 1. La structuration d'un document selon ODA

Pour faciliter la mise au point de documents, ODA permet aussi de définir des documents **génériques**, ou **classes** de documents (au sens des langages à objets). Un document particulier pourra alors être considéré comme une **instance** d'une classe donnée (lettre commerciale, rapport technique, bon de commande, thèse, etc.). Une classe est composée d'une structure logique générique, à laquelle peuvent être associées des portions de contenu génériques (formulaires par exemple), d'une ou plusieurs structures physiques

génériques et d'un style de document. Ce dernier précise des règles génériques pour passer d'une structure logique conforme à une des structures physiques possibles.

A côté de cela, ODIF définit un format standard d'échange de documents électroniques, c'est-à-dire un ensemble de règles pour la représentation des documents et des structures associées quand ils sont destinées à être transmis vers un autre système. Cela n'empêche pas bien sûr chaque système de stocker les documents comme bon lui semble de manière interne.

1.3 Le projet HERODE

Dans le cadre du programme de recherche **ESPRIT**[3], financé à 50% par la Communauté Economique Européenne, le projet **HERODE**[4], dont les deux principaux partenaires sont SIEMENS AG de Munich et TITN de Paris, a pour but de réaliser une station de bureautique intégrée permettant de traiter des documents conformes à la norme ECMA 101. Les composantes de ce système sont les suivantes :

- **L'éditeur de documents** ODA permet l'édition interactive avec formattage en ligne de documents composites. Il comprend des éditeurs spécialisés pour le texte, le graphique et le photographique, les trois types de contenu possibles dans un document. Il permet aussi de décrire de nouvelles classes et de manipuler la structure logique d'un document.
- **La saisie automatisée de documents** ou **ADE** (*Automated Document Entry*) a pour rôle de faire entrer un document sur papier dans l'univers électronique de ODA. Pour cela, ce module part d'une image saisie par une caméra haute résolution, la segmente en zones de contenus homogènes et code ceux-ci selon ODIF, pour qu'ils puissent être reprise par un système ODA. C'est le module dans lequel nous intervenons, et qui va être amplement présenté dans la suite de cette thèse.
- **L'archivage de documents** assure le stockage de documents sous leur forme électronique. Le support de stockage choisi est le **disque optique numérique**, qui a une capacité de 1 Gigaoctets, ce qui correspond à environ 500.000 pages de documents.
- **L'interface avec l'utilisateur** est un module chapeautant les autres et offrant ainsi une communication homme / machine harmonisée et uniforme. Il donne accès aux autres modules, par le moyen d'icônes représentant les outils disponibles à tout moment.

L'ensemble est implanté sur des stations de travail de type SUN, sous UNIX. Les langages de programmation choisis pour le projet sont C et **Objective-C**. Ce dernier est un un langage à objets [13] implantant au-dessus de C le mécanisme de classes et d'objets de **SMALLTALK-80**. L'emploi

---

3 European Strategic Programme for Research and Development in Information Technology

4 HERODE : Handling the Electronic Representation of Office Documents based on ECMA 101 standard

d'Objective-C est particulièrement judicieux pour la réalisation de toute la partie à forte interactivité (interface utilisateur, éditeurs). De plus, étant un sur-langage de C, il s'interface sans problème avec les modules écrits dans ce langage. Enfin, obligeant à bien préciser les objets manipulés, il a permis une intégration aisée des différents modules.

Dans le cadre de ce projet, le CRIN collabore à la réalisation de la partie ADE (saisie automatisée de documents), plus particulièrement sur la reconnaissance et le codage conformément à ODIF des zones graphiques dans un document composite. Dans toute la suite, notre propos portera donc sur cette partie. Le projet HERODE dans son ensemble est dans sa phase terminale, devant se conclure à l'automne 1987. L'état d'avancement des travaux a déjà été présenté à quelques occasions [28] ; un prototype de station de travail, où l'ensemble du logiciel développé aura été intégré, est prévu pour la fin de l'année.

1.4 Plan de cette thèse

Nous examinons d'abord les opérations de prétraitement d'images qu'il peut être nécessaire d'appliquer sur l'image brute. Ces opérations sont principalement des binarisations et des filtrages. Par la même occasion, nous passons rapidement en revue les principales techniques de codage et compression des images binaires. Nous ne présenterons pas les principes et le vocabulaire du traitement d'images, qui sont supposés connus pour le lecteur de cette thèse. On pourra toutefois se reporter au livre de Pratt [43] ou plus modestement à notre rapport de DEA [51].

Ensuite, un chapitre est consacré à la segmentation d'une image de documents. En effet, il est important de séparer les différentes zones informatives et de détecter pour chacune d'elles son type de contenu. Il est clair que les algorithmes à appliquer pour la reconnaissance de texte n'ont pas beaucoup de points communs avec ceux qui servent à reconnaître et à coder du graphique, par exemple. Nous présentons les principales méthodes de segmentation qui peuvent être employées et évaluons leurs avantages et inconvénients respectifs. Nous abordons aussi certains problèmes qui se retrouvent quelle que soit la méthode de segmentation employée, et qui souvent nécessitent d'autres genres d'informations, comme une connaissance *a priori* sur le contenu du document, pour être résolus de manière satisfaisante.

Le chapitre suivant fait le point sur les méthodes d'approximation polygonale. Nous entrons là dans la partie purement graphique, ces méthodes étant fondamentales si on désire prendre un peu de recul par rapport à l'image binaire et travailler plutôt sur des segments de droite, ou éventuellement des segments et des arcs de cercle. En effet, après présentation des principales familles d'algorithmes applicables, nous montrons qu'une approximation en polynômes de degré supérieur peut très bien être conçue comme une extension de l'approximation polygonale.

Il est alors naturel d'enchaîner sur l'extraction des lignes. Une figure géométrique étant presque toujours composée de structures linéaires, il est nécessaire de disposer de méthodes permettant l'extraction et le suivi de lignes. Dans ce domaine, nous sommes sortis des sentiers battus pour explorer une voie peu étudiée qui nous a semblé prometteuse.

Il est alors temps de présenter cette nouvelle méthode que nous avons développée. Pour cela, nous la replaçons dans son contexte en précisant les choix faits pour la réalisation des différentes étapes de l'ADE. Nous insistons surtout sur la structure de données qui est construite et enrichie au fil de ces différentes étapes, et qui nous permet de disposer d'une foule de renseignements une fois arrivés au niveau d'interprétation. Nous détaillons aussi la méthode d'extraction de lignes que nous avons conçue, ainsi que les étapes ultérieures de suivi de lignes et de recherche de courbes ouvertes et fermées.

Le module développé reconnaît des primitives graphiques élémentaires : courbes ouvertes et fermées. Toutefois, nous montrons au dernier chapitre que la structure fournie par ce module est assez riche pour permettre des interprétations de beaucoup plus haut niveau. Nous donnons quelques exemples de reconnaissance de primitives graphiques plus élaborées, et nous nous tournons vers les perspectives de développements ultérieurs, visant à la compréhension et l'analyse de plans techniques, par exemple.

Chapitre 2
Techniques de prétraitement

L'oeil est l'organe de la vision, mais le regard est acte de prévision et il est commandé par tout ce qui peut être vu, doit être vu et les négations correspondantes
Valéry

2.1 Introduction

2.1.1 Généralités

Le **traitement d'images par ordinateur** est un ensemble de techniques permettant d'appliquer un certain nombre de transformations à des images numérisées. Ces transformations peuvent avoir des buts très divers : filtrer l'image pour réduire le bruit, extraire des indices visuels particuliers pour une analyse ultérieure, segmenter l'image suivant un critère donné. En particulier, la segmentation est bien utile pour partitionner l'image en plusieurs régions homogènes, par exemple des régions d'intensité lumineuse constante, ou des régions ayant une propriété topologique ou géométrique particulière. Une **binarisation**, par exemple, peut très bien être considérée comme une segmentation en deux régions : le fond et la forme.

Dans l'abondante littérature traitant de ce sujet, on trouve un certain nombre de directions principales. Les premiers à s'intéresser au traitement d'images étaient au départ formés au traitement du signal. Ils considèrent donc très naturellement l'image comme un signal particulier, auquel ils appliquent des transformations classiques comme la transformée de Fourier (passage au domaine fréquentiel) ou des convolutions. D'autres transformations d'image, en particulier sur les images binaires, viennent des travaux sur le graphique (translations, rotations, zooms, ...). Une «école» intéressante est celle de la **morphologie mathématique** [48]. Fondée sur une théorie **topologique** de la forme, s'intéressant donc en premier lieu à **l'image binaire**, elle fournit un ensemble d'opérateurs permettant d'effectuer des filtrages morphologiques, et ayant des propriétés topologiques et ensemblistes bien définies.

2.1.2 Le cas des documents

Nous ne reprenons pas ici en détail l'étude faite en [51] sur les algorithmes de traitement d'images et en particulier sur la morphologie mathématique. Nous nous contentons de voir quels algorithmes sont utilisés dans le cas du traitement des documents.

Un document peut être numérisé par divers moyens. On utilise habituellement des caméras CCD haute résolution ou des *scanners* fournissant directement une image binaire. Si le document est censé pouvoir contenir des zones photographiques, il devient utile d'utiliser une caméra pouvant rendre une image à niveaux de gris. Toutefois, une autre solution dans ces cas serait d'employer un *scanner* qui rende une image **tramée**, c'est-à-dire composée de points noirs plus ou moins denses qui donnent une impression visuelle de niveaux de gris.

Il faut ici faire une distinction entre **l'information** et la **représentation** de cette information. Ainsi, une information binaire, c'est-à-dire une information de type **forme** ou **fond**, peut être aussi bien représentée par une image binaire (noir et blanc) que par une image à niveaux de gris. A l'inverse, une information à niveaux multiples, comme une photographie, peut être représentée par une image à niveaux de gris ou par une image binaire tramée. Dans les cas où le type de l'information ne correspond pas au type de la représentation, on retrouve l'opposition entre un niveau microscopique et un niveau macroscopique.

Dans ce chapitre, nous allons voir successivement les problèmes de binarisation d'image, puis de filtrage. Enfin, nous présentons quelques techniques de représentation et de codage d'une image binaire. Nous nous limitons volontairement aux spécificités des images de documents ; il est clair que pour d'autres types d'images, on peut trouver d'autres contraintes.

2.2 Binarisation

Quand l'image a été numérisée en niveaux de gris, il n'est pas judicieux de garder cette représentation pour les parties qui correspondent à une information binaire. Il est donc important de disposer de bonnes méthodes de **binarisation** d'une image.

Il est facile de binariser une image qui a un histogramme **bimodal**, c'est-à-dire présentant deux pics bien séparés. Il est alors relativement aisé de mettre en oeuvre une technique de recherche automatique du seuil, en analysant justement l'histogramme [58]. Mais quand l'histogramme est moins nettement séparable, une telle recherche de seuil devient beaucoup plus difficile. Certaines méthodes ont été proposées ; Pun par exemple propose de se servir de l'entropie de l'histogramme [44].

Cependant, même la meilleure méthode de sélection d'un seuil fixe est inadaptée si l'image de départ a été saisie avec un éclairage non uniforme, ayant entraîné des variations sensibles du fond, et parfois des reflets ou des zones en sur-brillance. Or, si certains *scanners* permettent de contrôler parfaitement l'éclairage, il n'en est pas de même d'une caméra qui bénéficie d'un éclairage d'appoint, difficile à régler, et pouvant subir les interférences des autres sources de lumière présentes dans le lieu d'acquisition (lumière naturelle variable, lampes, ombres portées par des objets ou des personnes, etc.). Il est donc souvent nécessaire de recourir à des techniques de **seuillage adaptatif**.

Le principe d'une telle binarisation est simple : il s'agit de ne pas fixer un seuil valable pour toute l'image, mais de faire varier ce seuil en fonction de l'aspect de l'image en différentes régions. Ainsi, Watson *et al.* [57] propose un algorithme appelé **double seuillage adaptatif** :

On fixe deux seuils, appelés `seuilHaut` et `seuilBas`, et un coefficient réel `coef` légèrement supérieur à 1. Le principe de l'algorithme est de distinguer les points faisant partie du fond, ceux qui appartiennent à des régions noires et ceux qui constituent des lignes noires. En chaque point, on calcule la moyenne de l'image sur un voisinage du point ; toutefois, on ne retient pour ce calcul que les points de valeur supérieure à `seuilBas`. Si le point a une valeur supérieure à cette moyenne pondérée par `coef`, il est considéré comme faisant partie d'une ligne noire. Sinon, si la valeur est supérieure à `seuilHaut`, le point est supposé faire partie d'une région noire. Enfin, dans les autres cas, il est mis à 0, car on suppose qu'il appartient au fond. L'algorithme peut alors s'écrire :

```
pour chaque point p de l'image
    moyenne = moyenne de tous les points de valeur supérieure à
              seuilBas dans un voisinage n x n de p

    si valeur(p) <= (moyenne * coef) alors
        si valeur(p) > seuilHaut alors
            res(p) = - valeur(p)
        sinon
            res(p) = 0
        fsi
    sinon
        res(p) = valeur(p)
    fsi
fpour
```

Les régions noires sont distinguées des lignes car elles correspondent à des points de valeur négative. Cet algorithme ne s'applique toutefois de manière satisfaisante que pour des épaisseurs de lignes à peu près connues ; une ligne trop épaisse par rapport aux paramètres choisis a de fortes chances d'être reconnue comme une région noire.

White et Rohrer présentent aussi deux algorithmes de binarisation intéressants [59]. Le premier, appelé **algorithme de seuillage dynamique**, est un peu analogue à celui que nous venons de présenter : le seuil

est déterminé par une moyenne dynamique calculée de manière récursive, la moyenne en un point dépendant de la valeur de l'image en ce point et de la moyenne des points déjà examinés, l'image étant parcourue par un balayage classique. Le calcul de cette moyenne dynamique se fait par une formule du genre

$$Moyenne_i = Moyenne_{i-1} + f(P_i - Moyenne_{i-1})$$

où f est soit une fonction linéaire de coefficient compris entre 0 et 1, soit une fonction non linéaire, mais vérifiant toujours $f(0) = 0$ et $0 \leqq f(x) \leqq x$.

Le second algorithme qu'ils proposent est plus spécifique à la reconnaissance de caractères, car il fait appel à une connaissance *a priori* sur la taille des zones noires à extraire. Il a été développé pour permettre d'extraire les zones binaires portant l'information même quand le fond est très tourmenté, comme c'est le cas pour l'écriture manuscrite sur un chèque par exemple. L'idée est de détecter les fortes variations d'intensité dans l'image, grâce à un gradient discret calculé sur une fenêtre de 3x3 points. Ces variations sont en effet potentiellement des contours de zones informatives. Pour éliminer les points contours ainsi détectés qui ne correspondent pas à une forme cherchée, on se sert de la connaissance *a priori* sur l'épaisseur normale d'un trait formant un caractère. Pour cela, on calcule un laplacien discret, en choisissant la largeur du masque en fonction de l'épaisseur de trait cherchée. La valeur et surtout le signe de ce laplacien permettent alors de différencier les vraies zones informatives du bruit. L'ensemble des opérations appliquées est combiné pour que tout le traitement se fasse en une seule passe sur l'image. Cette méthode donne des résultats remarquables sur des images de chèques par exemple ; cependant, elle dépend trop de connaissances préalables sur le type d'information qui existe dans le document pour pouvoir être appliquée telle quelle dans une binarisation quelconque. Néanmoins, les idées de cette méthode peuvent être exploitées ; c'est ce qui a été fait pour la binarisation mise en oeuvre dans le projet **HERODE**.

2.3 Filtrage de l'image

Le filtrage d'image est une technique employée à des fins diverses. Les filtres permettent d'éliminer certaines informations pour mettre en valeur celles qui restent. Cela est particulièrement utile pour **réduire le bruit** ; il reste évidemment à définir ce qui est bruit et ce qui ne l'est pas. Une technique simple consiste à remplacer chaque point par une moyenne pondérée calculée sur son voisinage. Des techniques plus évoluées permettent aussi de lisser l'image sans dégrader les contours, c'est-à-dire les zones de forte transition sur les niveaux de gris. Naturellement, l'élimination du bruit n'est qu'une application parmi d'autres du filtrage.

Dans le cas des images binaires de documents, c'est néanmoins le filtrage du bruit qui est l'application

principale, du moins à l'étape de prétraitement. Etant donné la nature de l'information portée par l'image binaire, il est souhaitable que ce filtre vérifie les contraintes suivantes :

- **Lissage des contours des formes.** Il est clair qu'un filtrage ne serait pas satisfaisant s'il donnait des régions binaires aux contours plus tourmentés que ceux de l'image de départ. Cela est particulièrement vrai quand on cherche des lignes dans l'image, comme c'est souvent le cas pour un document.
- **Préservation de l'information d'épaisseur** 1. En effet, il se peut très bien que l'image contienne des lignes très fines ; un filtre trop brutal les éliminerait. A l'inverse, si on garde tout ce qui est d'épaisseur 1 et au-dessus, on n'a plus un filtre mais la fonction identité!
- **Préservation de la connexité.** C'est une condition analogue à la précédente. Le principe est qu'une composante connexe de l'image d'origine sera éventuellement supprimée si elle est considérée comme inessentielle. Sinon, elle sera peut-être modifiée, mais pas divisée en deux composantes distinctes, sauf cas particulier (petit isthme joignant deux régions par ailleurs bien distinctes, par exemple).

A côté de ces propriétés liées au contenu sémantique de l'image, il est souhaitable que les méthodes développées soient rapides et aisées à mettre en oeuvre. Pour cela, certaines propriétés sont particulièrement intéressantes :

- **Efficacité** : un algorithme devrait de préférence ne nécessiter qu'**une passe** sur l'image, un nombre limité de lignes nécessaire en mémoire, des opérations simples et faciles à câbler.
- **Robustesse** : le filtrage doit pouvoir fonctionner de manière satisfaisante sur un grand nombre d'images, sans nécessiter systématiquement un ajustement des paramètres.
- **Stabilité** : il est préférable que l'algorithme soit **idempotent** ; en effet un filtre qui réduit de plus en plus l'information à chaque passe sur l'image a de fortes chances d'enlever trop de choses sur certaines images et pas assez sur d'autres. Un filtre destiné à enlever de petits bruits et qui ne serait pas idempotent enlèverait des informations non négligeables si l'image de départ se trouvait être déjà non bruitée. Le principe d'idempotence peut donc aussi s'énoncer : «enlever ce qu'il faut enlever, et rien d'autre».

La plupart des filtres vérifiant ces propriétés ou au moins certaines d'entre elles sont des opérateurs de voisinage. Une méthode très classique, par exemple, est de compter le nombre de points noirs et blancs dans un voisinage du point, de mettre un point noir à blanc s'il n'a pas assez de voisins noirs, et à l'inverse de mettre un point blanc à noir s'il a suffisamment de voisins blancs. Toutefois, si une telle méthode est efficace, robuste et stable, elle ne préserve pas forcément la connexité.

Un filtrage morphologique est à cet égard plus fiable. En particulier, les **épaississements** et les **amincissements** (*cf* [48]) sont bien adaptés. Ces opérations complémentaires mais non duales consistent à déterminer un certain nombre de voisinages pour lesquels un point noir passe à blanc (amincissement), et d'autres voisinages pour lesquels un point blanc passe à noir (épaississement). Si les voisinages sont bien choisis, la connexité est préservée. Nous donnons en figure 2 un ensemble de voisinages 3×3 permettant

de filtrer une image binaire en appliquant un certain nombre d'amincissements et d'épaississements successifs. Les **0** et les **1** indiquent un point appartenant respectivement au fond et à la forme, les points indiquent que la valeur est sans importance. Les huit configurations d'épaississement données correspondent au remplissage de petits «trous» dans des contours orientés dans les quatre directions cardinales et les quatre directions diagonales ; dans chaque cas, le **0** central est remplacé par un **1**. A l'inverse, les amincissements correspondent à l'effacement de «bosses» dans les huit directions, le **1** central passant à **0** :

Epaississements

```
. 0 .   1 . 0   1 1 .   1 1 1   1 1 1   1 1 1   . 1 1   0 . 1
1 0 1   1 0 .   1 0 0   1 0 .   1 0 1   . 0 1   0 0 1   . 0 1
1 1 1   1 1 1   1 1 .   1 . 0   . 0 .   0 . 1   . 1 1   1 1 1
```

Amincissements

```
. . .   0 . .   0 0 .   0 0 0   0 0 0   0 0 0   . 0 0   . . 0
0 1 0   0 1 .   0 1 .   0 1 .   0 1 0   . 1 0   . 1 0   . 1 0
0 0 0   0 0 0   0 0 .   0 . .   . . .   . . 0   . 0 0   0 0 0
```

Figure 2. Filtrages morphologiques

On remarquera que cette méthode préserve la connexité, sauf pour les régions noires qui se retrouvent réduites à un point isolé ; celles-ci sont supprimées. Le nombre d'itérations à appliquer peut être fixé une fois pour toutes ou rester un paramètre du programme. Cette méthode ne vérifie pas *a priori* la condition d'efficacité, dans la mesure où il est nécessaire d'opérer en plusieurs passes sur l'image. Toutefois, les opérations élémentaires sont très simples et l'ensemble peut facilement être «câblé», avec enchaînement des différentes passes grâce à des lignes de registres à décalage.

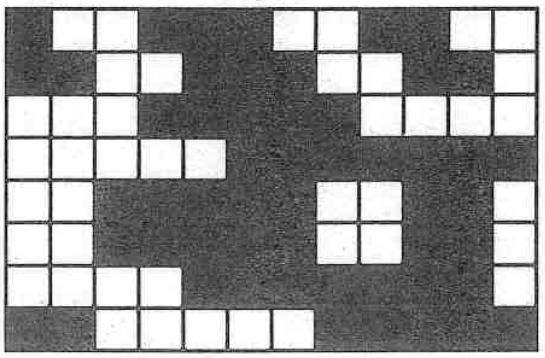
2.4 Compression et codage de l'image binaire

Le but de notre projet est d'interpréter le document saisi à un niveau relativement élevé ; par conséquent, le prétraitement est immédiatement suivi par d'autres modules de segmentation, de reconnaissance et de codage. Toutefois, dans beaucoup d'autres applications, il est inutile de mener une analyse plus poussée, le but étant tout simplement d'enregistrer ou éventuellement de transmettre sur une ligne l'image binaire résultant du prétraitement. C'est en particulier le cas des systèmes de **télécopie** (appelée aussi **fac-similé**). Le problème de la compression d'image se pose en fait plus généralement chaque fois qu'on désire transmettre une image ; Massip-Pailhes et Payrissat [32] entre autres présentent des applications dans des contextes variés, et non uniquement d'images de bureautique.

Nous passons en revue ici quelques-unes des méthodes usuelles de codage des images binaires ; Ehoud Ahronovitz présente dans sa thèse [1] une étude complète des algorithmes existants et de leurs caractéristiques respectives. Au niveau le plus élémentaire, on peut considérer bien sûr le codage de chaque point de l'image binaire par un *bit*. C'est déjà une représentation condensée par rapport à l'image de départ non binarisée, où chaque *pixel* est codé sur un octet dans la plupart des systèmes actuels.

Une autre technique bien connue est la division de l'image en zones plus homogènes. Ainsi, le principe d'un *quadtree* est de diviser récursivement l'image en 4 quadrants, et ceci jusqu'à obtenir des régions entièrement noires ou blanches. On crée ainsi une structure arborescente. Ce genre de méthode est parfois appliqué pour la compression d'images [26], mais présente l'inconvénient d'être très sensible au bruit et aux translations, même petites. Elle est peu pratique pour des images de documents typiques où il est fort probable de trouver beaucoup de texte, donc une répartition assez «aléatoire» des zones noires. Une autre technique plus adaptée aux documents est le codage par blocs [29] qui consiste à découper l'image en rectangles dont la taille est choisie de manière à ce qu'il y ait une forte probabilité d'avoir des rectangles entièrement blancs. Ceux-ci seront alors codés sur un seul *bit*, les autres étant codés comme une image binaire normale.

Un certain nombre de méthodes ont pour principe le **codage par plages**. Celui-ci peut être effectué en un seul balayage ligne par ligne de l'image. Dans sa forme la plus simple, il consiste à coder chaque ligne de l'image par les longueurs de ses plages noires et blanches successives (voir figure 3). Une image de texte contenant beaucoup de composantes connexes ayant sensiblement la même largeur, il peut être approprié d'utiliser un code de Huffman [24] pour la longueur des plages ; c'est le choix fait pour le code standard de télécopie du CCITT[5].



0 1 2 3 2 2 *
0 2 2 3 2 2 *
3 5 *
5 7 *
2 5 2 2 *
2 5 2 2 *
4 7 *
0 2 5 5 *

Figure 3. Codage par plages

5 Comité Consultatif International des Téléphones et Télégraphes

Une autre manière de coder une image binaire est d'en coder les contours. Le plus connu de ces **codages par contours** est celui de Freeman [18]. Il consiste à coder les huit directions cardinales par un nombre compris entre 0 et 7. Un autre codage analogue est celui proposé par Cederberg [12] ; il a l'avantage de pouvoir être créé en un unique parcours de l'image. La contrainte du parcours par balayage oblige toutefois à coder séparément les différentes branches descendantes du contour, puis à chaîner ces branches pour pouvoir manipuler le contour entier.

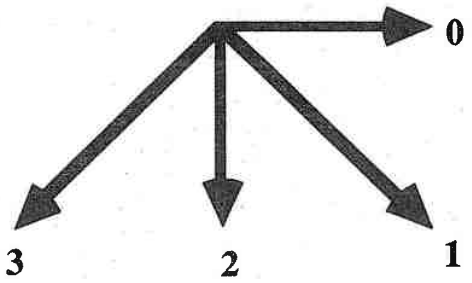


Figure 4. Codage par contours de Cederberg

La figure 4 illustre le principe de ce codage par contours. C'est la solution que nous avons adoptée pour coder l'images binaire dans le projet **HERODE**.

Il existe bien d'autres méthodes de codage, en particulier lorsqu'on utilise la connaissance *a priori* sur le contenu de l'image. A cet égard, Ahronovitz présente des algorithmes très efficaces. Mais comme notre but est de poursuivre l'analyse de l'image pour en extraire des primitives de plus haut niveau, nous nous contentons des méthodes exposées ci-dessus, puisqu'elles ne sont pour nous qu'un moyen provisoire de représenter l'image sur laquelle on travaille.

Chapitre 3

Segmentation d'un document

> Une syllabe n'a point de sens et même le mot n'en a guère. C'est la phrase qui explique le mot... les pages éclairées par ce regard d'exploration qui va de l'ensemble aux détails. Car enfin toute la page est vraie en même temps et il arrive souvent que la fin explique le commencement.
> Alain

La **segmentation d'images** est un sujet très vaste, sur lequel de nombreux travaux ont été menés. Segmenter une image consiste à trouver des régions ou des lignes de contours de l'image auxquelles on peut attribuer une étiquette **sémantique**, c'est-à-dire une signification concrète. Ainsi, pour la vision par ordinateur, il est souvent indispensable de trouver des régions homogènes en texture ou en intensité lumineuse, qui correspondent à des entités visuelles bien définies.

Le traitement des documents nécessite aussi des segmentations particulières [38]. C'est ce sujet que nous allons traiter dans ce qui suit, sans chercher à présenter de manière exhaustive le problème général de la segmentation d'images.

3.1 Contenu d'un document

Quand il est considéré d'un point de vue sémantique, un document est constitué d'un certain nombre de **zones informatives** homogènes. Au niveau le plus général, ces zones peuvent être classées en quatre grandes catégories :

- des zones **photographiques**, où l'information est essentiellement picturale et multi-niveaux ;
- des zones de **texte**, constituées de caractères regroupés en mots, phrases et paragraphes ;
- des zones de **graphique pur**, où se retrouvent des figures géométriques qui peuvent être interprétées en termes de primitives graphiques ;
- des zones **mixtes** texte / graphique qui sont en gros les régions où l'information est binaire mais difficile à identifier comme texte ou comme graphique, soit parce que les deux composantes y sont présentes de manière entremêlée au point de les rendre inextricables, soit parce que l'image est trop complexe pour être traitée comme un ensemble d'objets graphiques et de caractères (écriture manuscrite, signatures, logos, etc.).

Une première segmentation consiste à partager les zones photographiques, ou plus précisément les zones à

information multi-niveaux, des zones à information binaire. Nous présentons rapidement quelques idées pour effectuer cette segmentation. Nous détaillons ensuite beaucoup plus la segmentation de l'image binaire suivant le type de contenu. Pour cela, nous présentons les principales approches, avec leurs avantages et inconvénients respectifs.

3.2 Segmentation multi-niveaux / binaire

Il semble assez naturel de se fonder sur une **analyse d'histogramme** pour effectuer la segmentation entre les zones photographiques et celles contenant de l'information binaire. Toutefois, l'histogramme des niveaux de gris sur l'ensemble de l'image ne présente pas grand intérêt dans ce contexte, car il ne permet pas de détecter les zones correspondant effectivement à une photographie. La première idée est donc d'effectuer des analyses d'histogrammes locaux. En effet, sur une fenêtre relativement réduite, il est correct de supposer que l'histogramme est **bimodal** si l'information est binaire.

De plus, il est souvent nécessaire d'affiner le résultat, particulièrement aux frontières entre binaire et photographique. Une procédure récursive semble donc assez appropriée ; commençant par une segmentation relativement grossière fondée sur les histogrammes, elle s'applique récursivement avec une taille de fenêtre réduite sur les régions de contact entre zones binaires et zones multi-niveaux.

Il reste le problème des images tramées, où l'histogramme risque de ne pas donner grand-chose. Pour cela, Postl [41] propose de calculer sur chaque fenêtre la répartition statistique des points noirs et blancs, pour détecter d'éventuelles régions tramées. Ses travaux ultérieurs dans le cadre du projet **HERODE** le mènent à mettre également en oeuvre une **analyse de Fourier**, qui est coûteuse en temps mais qui, en passant dans le domaine fréquentiel, permet une segmentation plus fine [34]. Nous reviendrons sur le module de segmentation qu'il a développé pour notre projet. Dans une méthode que nous allons détailler ultérieurement, Wong, Casey et Wahl classent aussi certaines régions comme étant tramées ; cette segmentation se fait en même temps que celle entre texte et graphique.

Dans ce qui suit, nous supposons que la segmentation entre binaire et multi-niveaux a été effectuée ou que le document à traiter ne contient pas de photographies. Nous disposons donc d'une image binaire pouvant contenir du texte, du graphique et des zones mixtes.

3.3 Les lissages directionnels

Une méthode de segmentation a été proposée par Wong, Casey et Wahl [60]. Elle consiste à délimiter les zones informatives, puis à les analyser pour en déterminer le type. La méthode repose sur le lissage directionnel (*Run-Length Smoothing Algorithm* ou *RLSA*). C'est un algorithme qui, dans une direction donnée, comble les «petits» trous blancs pour former des blocs noirs continus. Pour la direction donnée, dans une ligne de points noirs et blancs, les points noirs sont laissés tels quels et les segments blancs ayant une longueur inférieure ou égale à un seuil C sont mis à noir. Ainsi, pour la direction horizontale, avec $C = 4$ et la séquence de points suivante :

0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0

la méthode donne :

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1

Remarquons au passage que c'est une **fermeture linéaire**, pour reprendre la terminologie de la morphologie mathématique. Ce lissage est appliqué horizontalement puis verticalement. Un **ET** logique des deux images ainsi obtenues donne un ensemble de blocs noirs qui sont les zones informatives du document. Sur chaque bloc ainsi délimité, les auteurs effectuent une analyse fondée sur les mesures suivantes :

- le nombre de points noirs du bloc (BC),
- sa taille,
- le nombre de pixels noirs correspondants dans l'image de départ (DC),
- le nombre de transitions noir / blanc dans l'image d'origine (TC).

Ces mesures permettent de déterminer les paramètres suivants :

- la hauteur $H = \Delta y$ du bloc,
- l'excentricité du rectangle englobant : $E = \Delta x / \Delta y$,
- le rapport du nombre de points noirs du bloc par rapport à sa surface : $S = BC / (\Delta x . \Delta y)$,
- la longueur moyenne des segments horizontaux dans l'image d'origine : $R_m = DC / TC$.

Une classification simple sur ces paramètres permet d'attribuer à chaque bloc un type de contenu. Quatre classes sont possibles :

1. Texte : le bloc contient des caractères organisés en mots, lignes et paragraphes,
2. Ligne horizontale,
3. Ligne verticale,
4. Image tramée ou graphique : c'est un peu la classe fourre-tout pour ce qui ne peut pas être classé ailleurs.

La dernière classe peut être soumise à une analyse plus fine pour distinguer des caractères et symboles isolés d'une part, du graphique pur d'autre part, et les images tramées éventuelles.

Trincklin [54], tout en utilisant le même genre de méthode pour délimiter les blocs, appuie ensuite l'analyse et la segmentation sur les caractéristiques des **blocs blancs maximaux** qui entourent les zones informatives.

Cependant, une grosse faiblesse des méthodes fondées sur les lissages directionnels est qu'elle privilégie les directions horizontale et verticale. Si le document, au moment de la saisie, est positionné de travers, même légèrement, les résultats sont faussés. Trincklin détecte l'inclinaison éventuelle et redresse le document en calculant l'angle entre la direction principale du document et l'horizontale. Cette méthode n'est cependant fiable que lorsque le document contient essentiellement du texte de taille *normale*.

Postl [42] propose une méthode plus coûteuse en temps mais aussi plus générale, puisqu'elle redresse aussi des images tramées par exemple ; il se fonde sur une analyse fréquentielle par l'intermédiaire d'une transformée de Fourier. La méthode consiste à intégrer la transformée de Fourier le long d'un vecteur d'angle β ; on fait varier β et on retient l'angle donnant une intégrale de valeur maximale. La détection peut être affinée par une recherche dichotomique de l'angle d'inclinaison quand on en a une première idée en ayant fait varier les angles de 5 en 5 degrés par exemple.

3.4 La segmentation par projections

Une autre méthode simple pour segmenter une image de document est l'étude des projections de l'image sur un axe horizontal et un axe vertical. Les lignes de texte, par exemple, donnent un pic sur l'axe vertical ; l'espace blanc séparant deux lignes dans un paragraphe correspond à une valeur nulle ou faible de la projection. L'application récursive de cette méthode aboutit à une segmentation du document en blocs rectangulaires organisés en arbre X-Y [39].

Ces blocs peuvent être analysés de différentes manières pour en déterminer le contenu ; Nagy, Seth et Stoddard [40] effectuent une analyse fondée sur un système à base de règles. Kida [27] analyse ces blocs en se fondant sur la taille des composantes connexes d'une part, pour extraire la plus grande partie des

caractères, et sur la répartition de la densité des points noirs d'autre part, pour retrouver les principaux objets graphiques ou mixtes.

Tout comme les méthodes fondées sur des lissages directionnels, l'utilisation de projections sur des axes n'est efficace que si le document traité n'a pas été saisi de travers, et que l'information dans ce document peut aussi être regroupée en blocs rectangulaires. Or, si une telle hypothèse peut presque toujours être faite quand le document à traiter contient essentiellement du texte[6], elle devient hasardeuse quand on est en présence d'une grande proportion de graphique ou de photos, comme c'est le cas entre autres pour beaucoup de documents techniques.

3.5 La segmentation par transformée de Fourier

Nous avons déjà présenté plusieurs utilisations de la transformée de Fourier, qui permet de passer dans le domaine fréquentiel, pour séparer l'information multi-niveaux de l'information binaire, ou pour redresser une image ayant été saisie de biais. Hase et Hoshino [22] proposent une méthode de segmentation entièrement fondée sur la transformée de Fourier dans l'espace à deux dimensions.

Ils commencent par effectuer une transformée de Fourier globale sur l'image, pour déterminer la direction principale du document, dans le cas où il a été saisi de travers, et la périodicité des lignes de texte. A partir de cette dernière information, on fixe une taille de fenêtre «raisonnable» et on effectue des transformées de Fourier locales dans des fenêtres de cette taille. L'analyse de l'information fréquentielle leur permet alors de déterminer le type du contenu dans les différentes régions. La méthode peut être affinée pour mieux fixer les frontières entre les régions de types différents.

Un reproche à faire à ce genre de méthode est bien sûr la complexité des calculs, dûs à la transformée de Fourier. Toutefois, si on dispose de circuits de calcul spécialisés, ce peut être une approche viable.

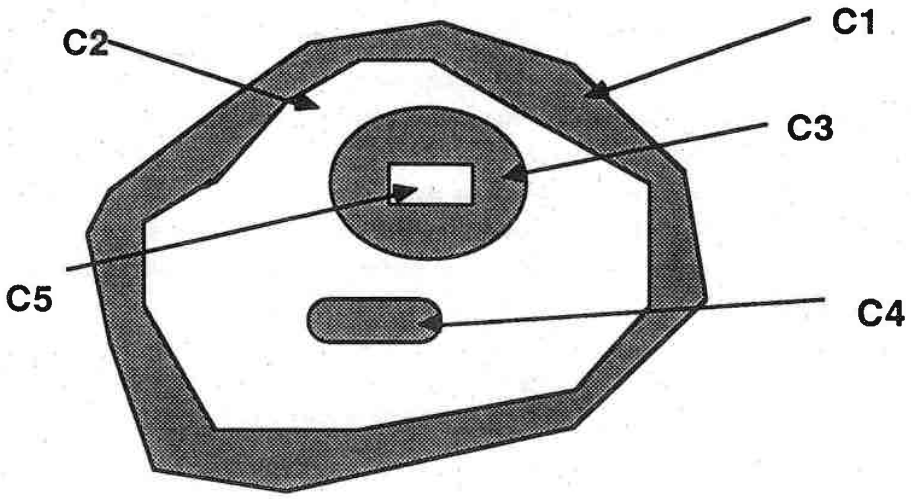
3.6 L'analyse des composantes connexes

Une image binaire contient un certain nombre de **composantes connexes** noires et blanches. Ces composantes ont des propriétés **intrinsèques**, comme leur taille, leur forme, leur compacité, etc. et des propriétés relationnelles entre elles (relations d'inclusion, de proximité...). L'extraction des composantes

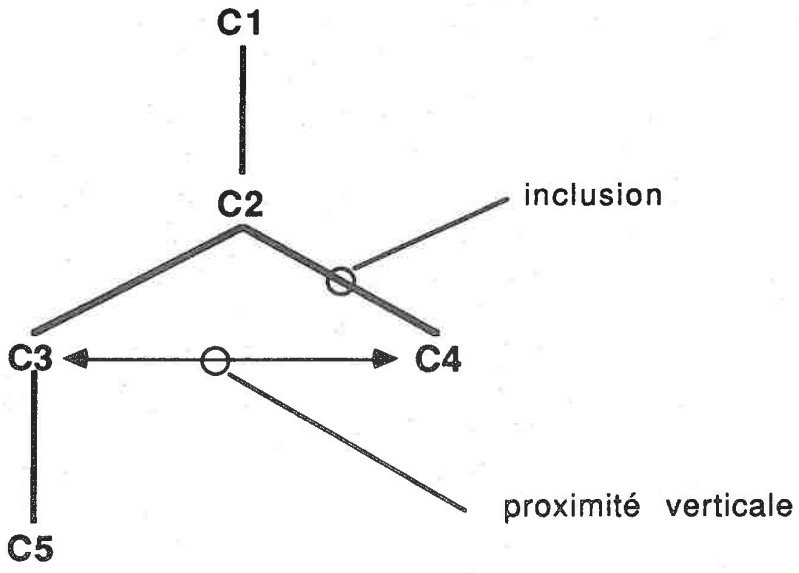
---

6 toutefois, il faut dans ce cas exclure de la catégorie «texte» les Calligrammes de Guillaume Apollinaire

connexes dans une image est une opération simple à mettre en oeuvre. Un seul parcours permet de construire un arbre de composantes connexes. La relation de père à fils dans cet arbre est une relation d'inclusion ; ainsi, une composante connexe noire peut avoir un certain nombre de composantes blanches incluses, et vice versa. Deux frères dans l'arbre correspondent à deux composantes de même couleur se trouvant au même niveau d'inclusion. On peut facilement ajouter des liens de proximité, par exemple, entre des composantes de même niveau. La figure 5 illustre la forme que peut prendre un tel arbre.



Une image binaire



La structure associée

Figure 5. L'arbre des composantes connexes

Comme une image de document contenant beaucoup de texte donnera un grand nombre de composantes

connexes, la segmentation par analyse de ces composantes risque d'être beaucoup plus coûteuse en temps de calcul que les autres méthodes. Cependant, c'est une méthode plus *fine* que les autres que nous avons présentées, et en particulier elle ne dépend *a priori* ni de l'orientation générale du document, ni de la taille des caractères. C'est pourquoi cette méthode est aussi proposée par certains, comme par exemple Doster [15]. Dans un article de synthèse sur l'analyse de documents, Srihari et Zack [49] citent comme critères d'analyse la taille des composantes, le nombre de segments dont elles sont composées, leurs propriétés topologiques et des mesures de leur forme.

La segmentation par composantes connexes a été choisie dans le projet **HERODE**, nous présenterons donc cette méthode avec plus de détails dans le chapitre présentant ce projet.

3.7 Quelques problèmes généraux de segmentation

Les différentes méthodes vues ont toutes des points forts et des points faibles. En particulier, plusieurs d'entre elles ne fonctionnent avec des résultats satisfaisants que si l'image n'est pas penchée et que l'information a bien deux directions principales, une horizontale et une verticale. Or, cela peut devenir douteux pour certains types de figures géométriques.

D'autres problèmes sont plus généraux. Cela est dû principalement au manque de connaissances **sémantiques** de toutes ces méthodes de segmentation. En effet, dans un contexte donné, un certain nombre de symboles peuvent porter une information de type «texte» alors que ces mêmes symboles seront plutôt reconnus comme des figures géométriques quelconques hors de ce contexte. De même, une ligne de tirets pourrait très bien être une colonne de lettres «l» ; là encore, seule une compréhension du contenu sémantique du document considéré permettrait de trancher définitivement.

Les gros caractères et les **logos** posent aussi des problèmes de ce genre : un très gros caractère reste-t-il un caractère, ou est-ce un objet graphique? Si c'est un caractère, sur quel critère peut-on le décider sans que d'autres objets purement graphiques se retrouvent par inadvertance classés comme caractères?

Un autre problème difficile est le cas où le texte «touche» le graphique. En fait, il semble à peu près impossible de régler cela à la segmentation ; les solutions proposées sont toujours appliquées plus tard, le texte lié au graphique étant dans un premier temps considéré comme graphique [19] [6]. C'est au moment du suivi de lignes par exemple que les caractères seront détectés comme des *excroissances* gênantes de celles-ci. Ils sont alors éliminés de la partie graphique et ajoutés à la partie texte.

Chapitre 4
L'approximation polygonale

> Puis, ayant pris garde que, pour les connaître, j'aurais quelquefois besoin de les considérer chacune en particulier, et quelquefois seulement de les retenir, ou de les comprendre plusieurs ensemble, je pensai que, pour les considérer mieux en particulier, je les devais supposer en des lignes, à cause que je ne trouvais rien de plus simple, ni que je pusse plus distinctement représenter à mon imagination et à mes sens...
> Descartes

4.1 Le problème

Un document graphique est constitué dans une large mesure de lignes et courbes, c'est-à-dire de formes binaires pour lesquelles les informations les plus utiles sont l'épaisseur et la ligne médiane. Aussi longtemps que la vision qu'on a de l'image n'est que celle d'une matrice de points noirs et blancs, ces structures linéaires ne sont pas aisées à déterminer. Il importe donc de «prendre du recul» par rapport à l'image d'origine pour «voir» des lignes ou du moins des portions de lignes. C'est le principe général de l'approximation polygonale, qui n'est elle-même qu'un cas particulier, mais de loin le plus commun, de l'approximation d'un ensemble de points par une courbe. Ce problème plus général est loin d'être définitivement réglé, et on trouve des travaux théoriques [4] [36] portant sur les critères à vérifier pour que les principales caractéristiques d'une courbe, et en particulier sa courbure, soient préservés par son approximation. De plus, pour les problèmes de reconnaissance des formes, il est souhaitable que la méthode d'approximation soit invariante par rotation, translation et homothétie, pour que l'orientation de la forme n'ait pas d'influence sur la reconnaissance.

Les idées qu'on trouve dans les algorithmes d'approximation sont souvent les duales de celles qui régissent les traitements de graphiques, où on se pose le problème inverse, à savoir **comment représenter une courbe mathématique par un ensemble de points sur une trame.**

Parmi les nombreux algorithmes d'approximation polygonale existants, on distingue principalement deux grandes familles. La première est fondée sur la notion de division récursive ; elle est très efficace pour l'approximation des contours d'une forme par un polygone décrivant cette forme de manière fiable. L'autre famille d'algorithmes repose sur le suivi séquentiel de la ligne à approximer. L'avantage de cette dernière méthode est de ne nécessiter qu'un seul parcours de la courbe, et de pouvoir éventuellement être effectuée

en un seul parcours de l'image. Nous présentons ci-après ces deux types d'algorithmes et détaillons plus particulièrement le second.

4.2 La division récursive

L'approximation polygonale par division récursive consiste à découper la courbe aussi longtemps que chaque portion ne peut être approchée par un segment de droite avec une précision satisfaisante. Nous donnons ci-dessous un exemple typique d'algorithme d'approximation par division récursive. Dans cet algorithme, considérons les deux extrémités d'une courbe ouverte ; il est entendu que si la courbe est fermée, ces deux extrémités seront confondues. Pour décider de la validité de l'approximation d'une portion de courbe par un segment, plusieurs critères peuvent être employés. Ici, nous prendrons celui de la plus grande distance entre un point de la courbe et le segment de droite ; soit *dmax* la distance maximale tolérée. C'est le critère cité par exemple par Ramer [45] et par Landy et Cohen [30].

La fonction d'approximation polygonale consiste alors simplement à calculer la distance maximale entre les points origine et extrémité de la courbe. Si cette distance est supérieure au seuil qu'on s'est fixé, on applique la fonction récursivement sur les deux sous-courbes délimitées en coupant la courbe au point le plus éloigné :

```
approx (origine, extrémité) :

Soit seg le segment [origine extrémité]

pointCoupe = origine;
distMax = 0;

pour chaque point p de la courbe de origine à extrémité
    si distance (p, seg) > distMax alors
       pointCoupe = p;
       distMax = distance (p, seg);
    fsi
fpour

si distMax > dmax alors
    retourner ( approx (origine, pointCoupe) U
                approx (pointCoupe, extrémité));
sinon
    retourner seg;
fsi
```

Un des principaux avantages de cette méthode est qu'elle préserve bien les points anguleux. En effet, comme on effectue les divisions aux points les plus éloignés de la courbe de départ, les points où la direction change brusquement seront des points de coupure privilégiés. Nous verrons que cela est beaucoup moins évident avec d'autres méthodes.

En revanche, la méthode de division récursive a deux gros inconvénients. Le premier est qu'il faut disposer de tout le contour en mémoire, puisqu'il est parcouru dans sa totalité à chaque étape de l'algorithme. Cela est contraignant lorsqu'on veut effectuer l'approximation d'une image binaire, qui peut être de très grande taille, ou qu'on veut intégrer l'opération d'approximation dans une chaîne de traitements éventuellement «câblés». Le second inconvénient de la division récursive est la complexité de l'algorithme : supposons que la courbe à approcher est homogène (c'est-à-dire qu'elle ne présente pas de parties plus «tourmentées» que d'autres) et que le résultat final est un ensemble de n segments. L'approximation se fait alors en $\log n$ étapes de récurrence, et par conséquent la courbe entière sera parcourue $\log n$ fois, alors que d'autres méthodes, comme celles que nous allons exposer par la suite, effectuent le travail en un seul parcours de la courbe.

Il est souvent fait référence à la faiblesse d'un algorithme comme celui-ci pour la prise en compte des points aberrants. En effet, si un point du contour présente une déviation brutale et ponctuelle par rapport à la direction générale, la méthode va le détecter et renforcer son influence, au lieu de l'éliminer. Ce problème se pose parfois dans des saisies à la tablette graphique par exemple, où des erreurs d'échantillonnage ou même une démagnétisation locale peuvent susciter de telles aberrations. Si de tels points aberrants doivent effectivement être supprimés, il faut ajouter un module spécifique, qui détecte par exemple, après l'approximation, les paires de segments correspondant à une aire très faible. Toutefois, nous n'attachons pas grande importance à ce genre de problème dans la présente étude ; en effet, il est raisonnable de se fier aux étapes précédentes de numérisation du document et de prétraitement de l'image, pour supposer que tout pic de la courbe, même isolé, est important et doit être pris en compte s'il n'a pas déjà été éliminé au cours de ces étapes.

Cependant, la méthode de division récursive garde de sérieux atouts. Outre sa simplicité conceptuelle, elle a l'avantage de donner des approximations de bonne qualité sans qu'il ne soit nécessaire de poser des hypothèses particulières sur le type de courbe à traiter. Elle est en particulier souvent utilisée dans des problèmes de reconnaissance des formes, où il importe d'approcher une forme par un polygone de manière à bien isoler les points caractéristiques du contour de la forme.

4.3 Le suivi séquentiel

Une autre grande famille d'algorithmes est celle des méthodes de suivi séquentiel. Le principe est d'engendrer une suite de segments par un seul parcours de la courbe à approcher. Cette courbe peut exister

déjà sous forme d'image binaire, ou être créée de manière interactive, par exemple au moyen d'une tablette graphique. Dans ce dernier cas, l'approximation polygonale devra aussi se faire «à la volée», au fur et à mesure du tracé de la courbe.

Un problème inhérent à ces méthodes de suivi séquentiel est la vision purement locale qu'elles ont de la courbe à approcher, ce qui rend plus difficile la détection des points caractéristiques de la courbe. On risque en particulier de mal placer les points anguleux, et ainsi de déformer la figure traitée. L'exemple classique est le traitement de l'angle droit d'un rectangle : au moment où il est positionné sur cet angle, l'algorithme n'a aucun moyen de savoir s'il correspond à une petite variation à éliminer ou à un vrai changement de direction. Nous verrons que pour pallier cet inconvénient, il est possible d'implanter des méthodes de retour arrière qui «redressent les angles» en particulier.

Un élément important de toute méthode d'approximation est le critère sur lequel on décide d'accepter ou non un segment comme une bonne approximation d'une portion de courbe. Passons en revue les principaux critères employés, en détaillant plus longuement la méthode que nous avons décidé de mettre en oeuvre.

4.3.1 Les secteurs angulaires

Pour la division récursive, nous avons vu qu'un bon critère pouvait être la distance entre le segment et le point de la courbe qui en est le plus éloigné. Ce critère peut aussi être employé dans le cas du suivi séquentiel ; seulement, si on revient à chaque fois sur tous les points intermédiaires pour calculer la nouvelle distance au segment de droite, l'intérêt du suivi séquentiel disparaît, puisqu'il faut parcourir plusieurs fois chaque portion de courbe.

Dunham [16] règle ce problème par le moyen des **secteurs angulaires** : soit un ensemble de points $P_0 \ldots P_n$ et une distance maximale *dmax* admise entre un point de la courbe et le segment d'approximation. Au point P_1, toute droite passant dans le secteur angulaire défini par le point P_0 et le disque de rayon *dmax* centré en P_n est à une distance inférieure à *dmax* du point P_1. De même, au point P_2 pour le secteur angulaire défini par P_0 et le disque centré en P_2. L'ensemble des droites passant à une distance inférieure à *dmax* de P_1 et de P_2 est donc l'intersection de ces deux secteurs angulaires. La méthode consiste donc à calculer pour chaque point de la courbe le secteur angulaire associé, et à poursuivre le segment courant tant que l'intersection de tous ces secteurs angulaires reste non vide, ce qui garantit que tous les points seront suffisamment proches du segment d'approximation. Quand l'approximation est arrêtée, on peut prendre par exemple la bissectrice du secteur angulaire restant comme segment d'approximation. Cette méthode est schématisée par la figure 6.

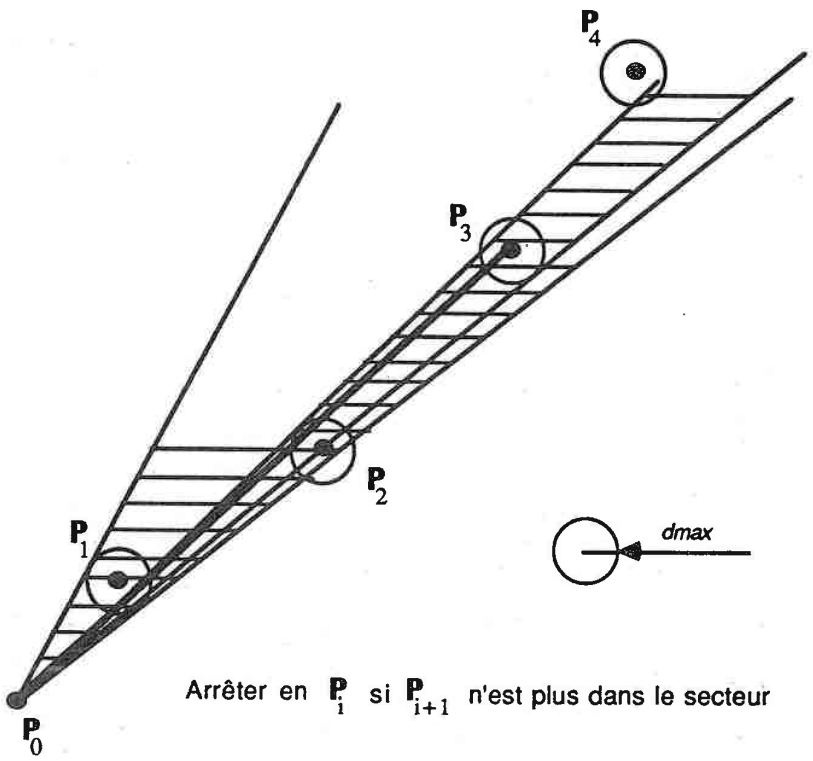


Figure 6. Approximation polygonale par secteurs angulaires

4.3.2 Les calculs d'angle

De nombreux autres algorithmes utilisent des calculs d'angle pour décider de la validité d'une approximation. Cela permet dans une large mesure de préserver les angles, de ne pas les déplacer. En se fondant sur les travaux de Berthod et Jancenne [8], Belaïd et Masini [7] ont implanté une méthode d'approximation de courbes tracées à la tablette graphique ; le principe est de retenir un point comme début d'un nouveau segment et fin du segment courant si ce dernier a une longueur suffisante et que l'angle entre la direction courante du segment et la direction locale de la courbe est supérieur à un seuil donné (voir figure 7). Leur algorithme regroupe aussi des groupes de segments en primitives du genre «ligne droite», «portion circulaire», ou en combinaisons de ces primitives.

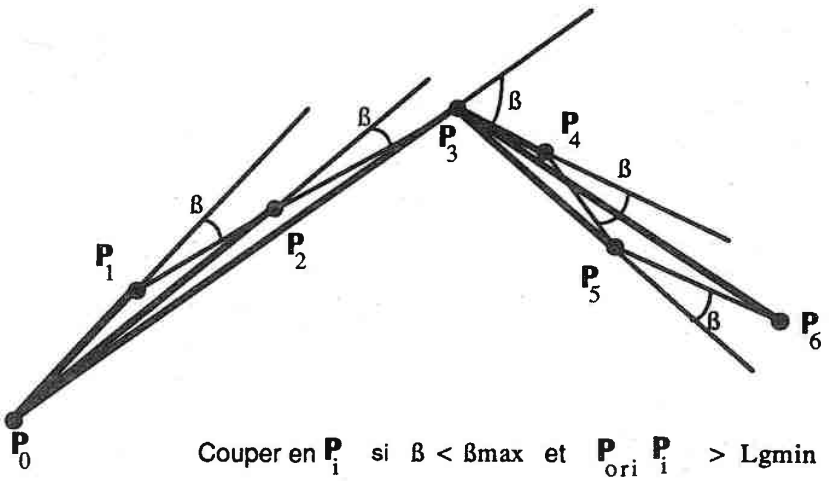


Figure 7. Approximation de tracés par calcul d'angles

4.3.3 La méthode de Wall et Danielsson

Wall et Danielsson [56] proposent d'utiliser l'aire algébrique comprise entre la courbe et le segment qui l'approche comme critère d'arrêt pour l'approximation. Ils fournissent pour la calculer un algorithme très simple, qui permet à la méthode d'être tout particulièrement efficace et rapide.

Supposons qu'on veut approcher la courbe formée par les points $P_0, P_1, \ldots, P_n$. On se ramène par une simple translation à un repère où l'origine est en P_0, ce qui revient à travailler en coordonnées relatives par rapport à P_0. L'algorithme consiste en quelque sorte à effectuer une intégrale curviligne le long de la courbe. A une étape k donnée, la somme des aires algébriques des triangles $[P_0P_{i-1}P_i]$ pour i variant de 1 à k (voir figure 8) est égale à l'aire algébrique comprise entre le segment $[P_0P_k]$ et la courbe. On arrêtera le segment courant quand cette aire algébrique devient trop grande. Le seul paramètre de la méthode est un seuil T, dont la signification sera précisée par la suite. L'algorithme d'approximation s'écrit alors comme suit :

- On part avec le segment $[P_0\ P_1]$, qui est bien sûr une approximation exacte de la courbe entre ces deux points. On pose $f_1 = 0$.
- A l'étape i, soient x_i, y_i les paramètres de P_i et Δx_i, Δy_i les variations en x et y pour passer du point P_{i-1} au point P_i. On calcule :

$$f_i = f_{i-1} + \Delta f_i \quad \text{avec} \quad \Delta f_i = x_i \cdot \Delta y_i - y_i \cdot \Delta x_i$$

Il faut noter que par cette formule, c'est f_i / 2 qui est égal à l'aire algébrique entre la courbe et le segment. L'avantage est qu'avec des points de coordonnées entières, f_i reste toujours entier.

- Calculer la longueur du segment $[P_0P_i]$, par exemple dans la métrique euclidienne classique :

$$L_i = \sqrt{x_i^2 + y_i^2}$$

On pourrait aussi, bien sûr, employer une autre métrique qui permette de travailler sur des entiers, mais les métriques carrées usuelles, d_1 et d_∞, ne sont pas isotropes, et nous allons voir qu'on peut rester en arithmétique entière par un autre moyen.

- Le segment est une approximation valable si :

$$|f_i| \leqq T \cdot L_i$$

Si ce test n'est pas vérifié, on arrête le segment au point P_{i-1} et on démarre un nouveau segment qui pour l'instant est réduit à $[P_{i-1}P_i]$.

Il faut noter que pour k donné, f_k est égal au double de l'aire algébrique entre la courbe et le segment $[P_0P_k]$. Le seuil T permet de faire varier la précision de l'approximation. On remarquera par ailleurs que le critère d'arrêt n'est pas une valeur maximale admise pour l'aire, mais que plus le segment est long, plus l'aire pourra aussi être importante sans pour autant invalider l'approximation.

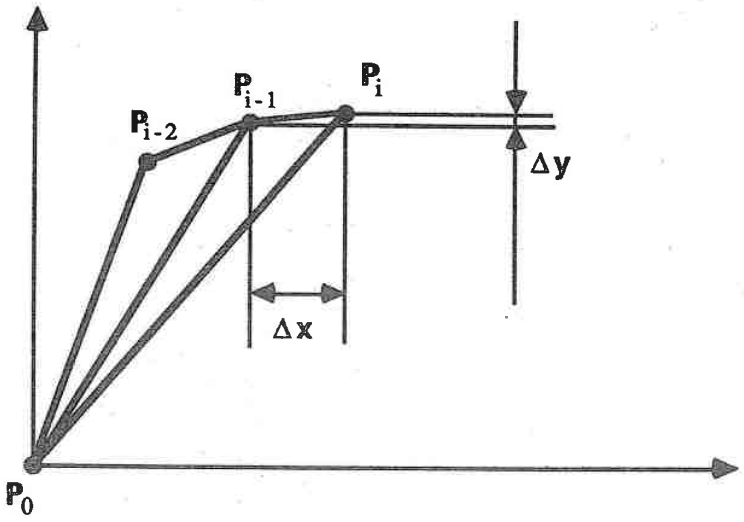


Figure 8. La méthode de Wall et Danielsson

Cette méthode permet l'approximation d'une courbe en seul parcours. De plus, on peut remarquer que le test d'arrêt peut aussi s'écrire :

$$f_i^2 \leqq T^2 \cdot L_i^2$$

Tous les calculs deviennent alors **entiers** quand les points ont des coordonnées entières ; cela donne un programme très rapide par rapport aux autres méthodes que nous avons vues.

4.4 Préservation des points anguleux

Plusieurs méthodes d'approximation polygonale ont l'inconvénient de ne pas préserver les points anguleux. En particulier, la méthode de Wall et Danielsson présentée ci-dessus *rogne les coins* si elle est appliquée telle quelle, comme l'illustre la figure 9. En effet, la vision restant très locale, on ne sait pas au moment où on détecte un changement de direction si celui-ci n'est dû qu'à une perturbation de la courbe à approcher, ou s'il correspond réellement à une nouvelle direction générale. Ce n'est qu'un peu plus loin, au moment où l'aire devient trop importante, qu'on décide d'arrêter le segment courant et d'en commencer un nouveau. Malheureusement, le point anguleux a déjà été dépassé à ce moment-là.

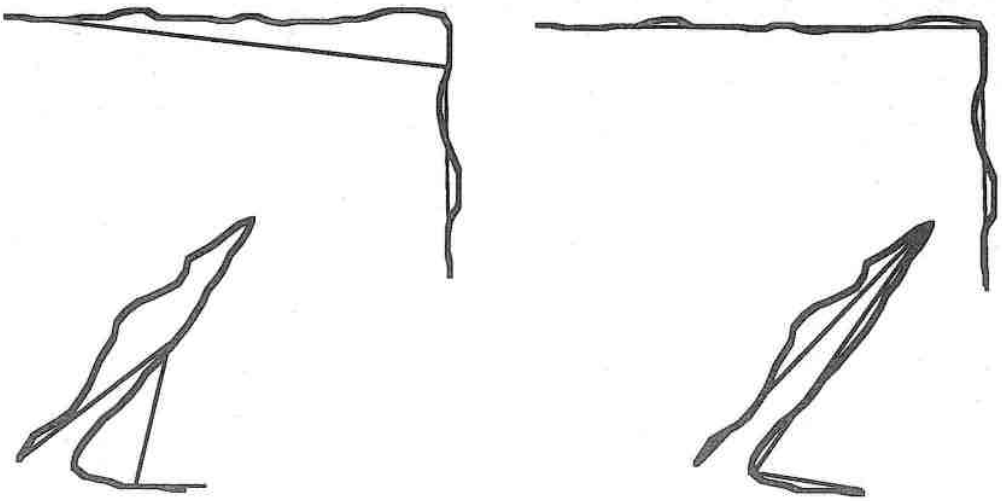


Figure 9. Non préservation des points anguleux

Il y a donc deux questions différentes à se poser : d'abord **quand couper**, et ensuite **où couper**. On pourrait évidemment résoudre ce problème en revenant sur l'ensemble de la courbe approchée et en trouvant le point le plus éloigné du segment par exemple. Répétons qu'une telle solution n'est pas entièrement

satisfaisante dans la mesure où le principe du suivi séquentiel est justement de ne suivre la courbe qu'une seule fois.

Dans le cadre de notre projet, nous avons donc développé une autre méthode permettant de préserver les points anguleux. Le principe est de marquer comme angle potentiel un point en lequel la direction de la courbe change de manière notable. Si la courbe reprend sa direction habituelle, la marque est effacée, le changement de direction n'étant qu'une perturbation locale. Quand il faut arrêter la courbe à cause du critère de Wall et Danielsson, on l'arrêtera au point marqué éventuel et non au point courant.

Nous donnons ci-dessous un algorithme général d'approximation polygonale avec préservation des angles droits, utilisant la méthode de Wall et Danielsson comme critère d'approximation proprement dit. Nous considérons la courbe $C = \{ P(i) \; pour i \in [0 \ldots n] \}$. L'algorithme crée un ensemble de segments chaînés ; à chaque segment est associée, entre autres paramètres, une direction générale codée sur un octet, suivant les directions cardinales (type codage de *Freeman*).

Le principe que nous avons employé est que lorsque la direction empruntée pour aller au point suivant est différente de la direction générale du segment courant, le point courant est marqué comme point de coupure potentiel. Si la direction revient à la normale, ce n'était qu'un «accident de parcours» et la marque est effacée. En revanche, si le changement de direction persiste, l'approximation finit bien par être arrêtée sur le critère de Wall et Danielsson. On arrête alors le segment courant au point marqué et non au point courant ; c'est aussi à partir de ce point de coupure qu'est démarré le nouveau segment courant :

```
segmentCourant = [P(0) P(1)];
f = 0;
dirGen = direction générale de [P(0) P(1)];

pour i de 2 à n
    calculer critère de Wall et Danielsson sur P(i);

    si P(i) peut être ajouté à segmentCourant alors
        changDirec = | dirGen - direction de [P(i-1) P(i)] |;
       si changDirec > 0 alors
            si il n'existe pas déjà un point de coupure cutP alors
                cutP = P(i-1);
                /*marquer P(i-1) comme point de coupure*/
            fsi
        sinon
            si on avait marqué un point de coupure cutP alors
                cutP = nil;
                /*effacer la marque*/
            fsi
       fsi

        P(i) est la nouvelle extrémité de segmentCourant;
        f = nouvelle valeur calculée par le critère Wall-Danielsson;
       dirGen = direction générale de segmentCourant;

    sinon
        si on avait marqué un point de coupure cutP alors
           arrêter segmentCourant à cutP;
           segmentCourant = [cutP P(i)];
           mise à jour de f;
           /*vérifier éventuellement que ce nouveau segment*/
           /*courant reste valide au sens Wall et Danielsson*/
        sinon
           arrêter segmentCourant à P(i-1);
           segmentCourant = [P(i-1) P(i)];
           f = 0;
       fsi

       mise à jour de dirGen;
    fsi
fpour
```

Il faut noter que pour mettre à jour facilement la mesure `f` de l'aire algébrique dans le cas où on arrête le segment courant au point de coupure, il faut aussi gérer un segment que nous appelons **segment de coupure**, et qui correspond à tout moment à `[cutP P(i)]`. Sa direction générale et l'aire `f` associée étant

mises à jour à chaque progression sur la courbe, dans la mesure où il existe un point de coupure marqué, la reprise sur ce nouveau segment à l'arrêt du segment courant est immédiate.

4.5 Extension vers l'approximation en segments et arcs de cercle

Nous avons dit au début de ce chapitre que l'approximation polygonale n'est qu'un cas particulier de l'approximation d'une courbe par une fonction du $n^{ème}$ degré. En particulier, il peut être intéressant de faire une approximation en **segments de droite et en arcs de cercle**. En effet, une faiblesse de l'approximation polygonale est qu'elle déforme et morcelle les arcs circulaires, ce qui complique à un niveau plus élevé la reconnaissance de primitives graphiques du genre **cercle** ou **ellipse**.

Un certain nombre de publications traitent de ce problème comme d'une extension de l'approximation polygonale. En particulier, Alami *et al.* [2] proposent d'effectuer d'abord une approximation polygonale, puis de voir, pour chaque segment de cette approximation, si un arc de cercle centré sur la médiatrice du segment n'est pas une meilleure approximation de la courbe.

En se fondant sur la division récursive, on peut aussi concevoir une extension simple de l'approximation en arcs de cercle, en calculant l'arc de cercle joignant trois points non alignés. Il suffit pour cela d'opérer en deux étapes : si l'approximation par un segment est bonne, on la retient ; sinon, on cherche à approcher la courbe par un arc de cercle joignant les deux extrémités et le point le plus éloigné du segment passant par celles-ci. Ce n'est que si cette approximation est également trop mauvaise qu'on répète le procédé récursivement sur les deux sous-courbes délimitées. Voici un algorithme modifié en conséquence :

```
approx2 (origine, extrémité) :

Soit seg le segment [origine extrémité]

pointCoupe = origine;
distMax = 0;

pour chaque point p de la courbe de origine à extrémité
    si distance (p, seg) > distMax alors
       pointCoupe = p;
       distMax = distance (p, seg);
    fsi
fpour

si distMax > dmax alors
    arc = arc de cercle (origine, pointCoupe, extrémité)
    si arc est une bonne approximation alors
        retourner arc
    sinon
        retourner ( approx2 (origine, pointCoupe) U
                    approx2 (pointCoupe, extrémité));
    fsi
sinon
    retourner seg;
fsi
```

Une telle méthode entraîne évidemment des calculs plus complexes, aussi bien pour déterminer un arc de cercle à partir de trois points que pour calculer la *distance* de cet arc de cercle à la courbe.

Il est aussi intéressant de noter l'article de Karin Wall [55], qui montre que la méthode de suivi séquentiel que nous avons appelée méthode de Wall et Danielsson peut être étendue à une approximation en polynômes de degré supérieur à 1. En partant de l'approximation polygonale, on détermine un polynôme de Hermite [17] qui est une bonne approximation de la courbe. La méthode ainsi définie crée des portions de courbe se raccordant sans variations inconsidérées de la tangente. De plus, le polynôme de Hermite est choisi de manière à retrouver l'aire algébrique fournie par l'approximation polygonale.

Malgré l'intérêt de ces extensions à des courbes de degré supérieur, nous ne les avons pas implantées dans la version actuelle de notre système, principalement à cause du surcoût en temps de calcul qu'elles induisent. Cependant, la possibilité d'ajouter de telles méthodes subsiste, et nous permet de garder confiance en l'approximation polygonale comme méthode de compression et d'extraction de l'information, puisque ses défauts constatés sur des portions circulaires de courbes peuvent être palliés sans remettre en cause les principes de base.

Chapitre 5
L'extraction de lignes

> On n'a pas trop de toutes ses forces de soumission au réel, pour arriver à faire passer l'impression la plus simple en apparence du monde de l'invisible dans celui si différent du concret où l'ineffable se résout en claires formules.
> Proust

5.1 Le problème

Nous avons déjà dit que l'information graphique dans un document est principalement portée par des **structures linéaires**, et qu'il importe donc de retrouver celles-ci dans l'ensemble des parties graphiques. Cependant, ce ne sont pas les seules composantes à prendre en considération : il existe aussi des zones noires ou blanches qui ne représentent pas une ligne, mais une forme quelconque. Dans ce cas, c'est le contour de la zone qui porte l'information.

La séparation entre ces deux types d'objets graphiques ne peut être qu'arbitraire. Il n'existe pas de critère aveugle permettant de dire qu'un objet est une zone noire ou une ligne. Cette ambiguïté est illustrée par la figure 10 : avec notre vision globale, nous aurons tendance à considérer le même objet graphique comme une ligne dans un cas et comme une tache noire dans un autre, suivant le contexte général de l'image.

Figure 10. Ligne ou tache noire?

Il se pose donc deux problèmes dans le traitement des parties graphiques d'un document. Le premier est de séparer les structures linéaires de ce qui est considéré comme taches noires. Le second est l'extraction des

lignes ; cela consiste à suivre les structures linéaires en retenant leur épaisseur, et à trouver les contours des taches noires. Nous allons présenter deux approches pour réaliser cette extraction ; la première est l'approche classique, choisie à quelques variantes près par presque tous les systèmes dont nous avons connaissance. La deuxième approche est plus originale, et n'a pour ainsi dire jamais été considérée avant. C'est celle que nous avons finalement choisi d'explorer, et nous présentons notre méthode en détail au chapitre suivant.

5.2 L'utilisation du squelette

La grande majorité des systèmes de reconnaissance de graphiques utilise d'une manière ou d'une autre la technique de **squelettisation**. Celle-ci consiste à trouver **l'axe médian** de l'image, qui sera considéré comme l'emplacement de la ligne graphique. Une propriété fondamentale du squelette est qu'il préserve la connexité de l'image. Il est pour cela souvent utilisé dans des problèmes de reconnaissance de formes, car il est un bon descripteur de forme [46].

Il existe beaucoup de méthodes pour déterminer le squelette ; on trouve d'abord celles qui sont fondées sur la détection de tous les points centres de disques maximaux contenus dans l'image [47], ce qui peut entre autres être fait par le calcul d'une transformée de distance [3]. Toutefois, ces méthodes peuvent être difficiles à appliquer à des images de taille importante, quand elles nécessitent par exemple des parcours en balayage inverse.

Une autre approche très souvent employée est celle des **amincissements** successifs. Le principe est de détecter des configurations où le fait d'enlever un *pixel* n'altère pas la connexité de l'ensemble, et d'itérer jusqu'à ce qu'on ne puisse plus enlever de points. Cette méthode est particulièrement bien adaptée aux systèmes «câblés», car les passes multiples peuvent être faites très rapidement et s'enchaîner dans un *pipe-line*.

Parmi les nombreuses applications de la squelettisation à l'analyse de documents, citons d'abord Landy et Cohen [30], qui calculent le squelette par passes successives sur l'image, avant de faire une approximation polygonale, par la méthode de division récursive, du squelette obtenu. Groen et van Munster [19] font aussi une squelettisation, mais sans approximation polygonale après, les résultats qu'il cherchent étant plutôt du domaine de la reconnaissance de symboles dans des schémas électriques et de leurs interconnexions.

Harris, Kittler *et al.* [21] ont une approche un peu plus originale : par la gestion d'une structure de données astucieuse, ils squelettisent l'image en une seule passe. Pour cela, ils gardent en mémoire un nombre variable de lignes ; une ligne sort définitivement quand on ne peut plus rien enlever. Avant de sortir

définitivement, le contenu de la ligne permet de mettre à jour un ensemble de vecteurs courants, par une approximation polygonale de type suivi séquentiel.

Toutefois, aucun de ces auteurs ne tient compte d'éventuelles taches noires pour lesquelles c'est le contour qui porte l'information utile. Dans une première étape du projet **HERODE**, nous nous étions aussi intéressés à l'utilisation du squelette. Présentons les grandes lignes de notre méthode, qui est détaillée en [52] :

- Pour séparer les taches noires des lignes, sur un seuil d'épaisseur n tout à fait arbitraire, nous effectuons une ouverture (c'est-à-dire une érosion suivie d'une dilatation) par un disque de taille n x n. L'avantage de l'ouverture est qu'elle élimine toute partie d'épaisseur inférieure à n sans modifier de manière notable le reste. Elle supprime donc les lignes d'épaisseur inférieure à n et lisse les taches noires. C'est sur cette image de taches lissées que nous effectuons une extraction de contours, ces contours étant intégrés par la suite à l'information linéaire extraite par ailleurs.
- L'image de taches lissée est soumise ensuite à une dilatation conditionnelle pour qu'elle retrouve sa forme d'origine. La différence avec l'image originale nous donne alors les lignes.
- Ces lignes sont squelettisées. Nous avons choisi d'implanter des amincissements successifs, les configurations suivantes permettant d'enlever un point sans changer la connexité :

```
0 0 0   1 . 0   1 1 1   0 . 1   . 1 .   . 1 .   0 0 .   . 0 0
. 1 .   1 1 0   . 1 .   0 1 1   1 1 0   0 1 1   0 1 1   1 1 0
1 1 1   1 . 0   0 0 0   0 . 1   . 0 0   0 0 .   . 1 .   . 1 .
```

- La réunion des contours d'un côté et du squelette des lignes de l'autre est soumise à une approximation polygonale par la méthode de Wall et Danielsson, avec un buffer de seulement trois lignes en mémoire à tout moment.
- Il faut évidemment garder une marque différente sur les contours et sur les lignes, et aussi préserver sur chaque point du squelette l'information sur l'épaisseur du trait d'origine. Cela n'est pas bien difficile à réaliser par des marquages *ad hoc* au moment de la squelettisation.

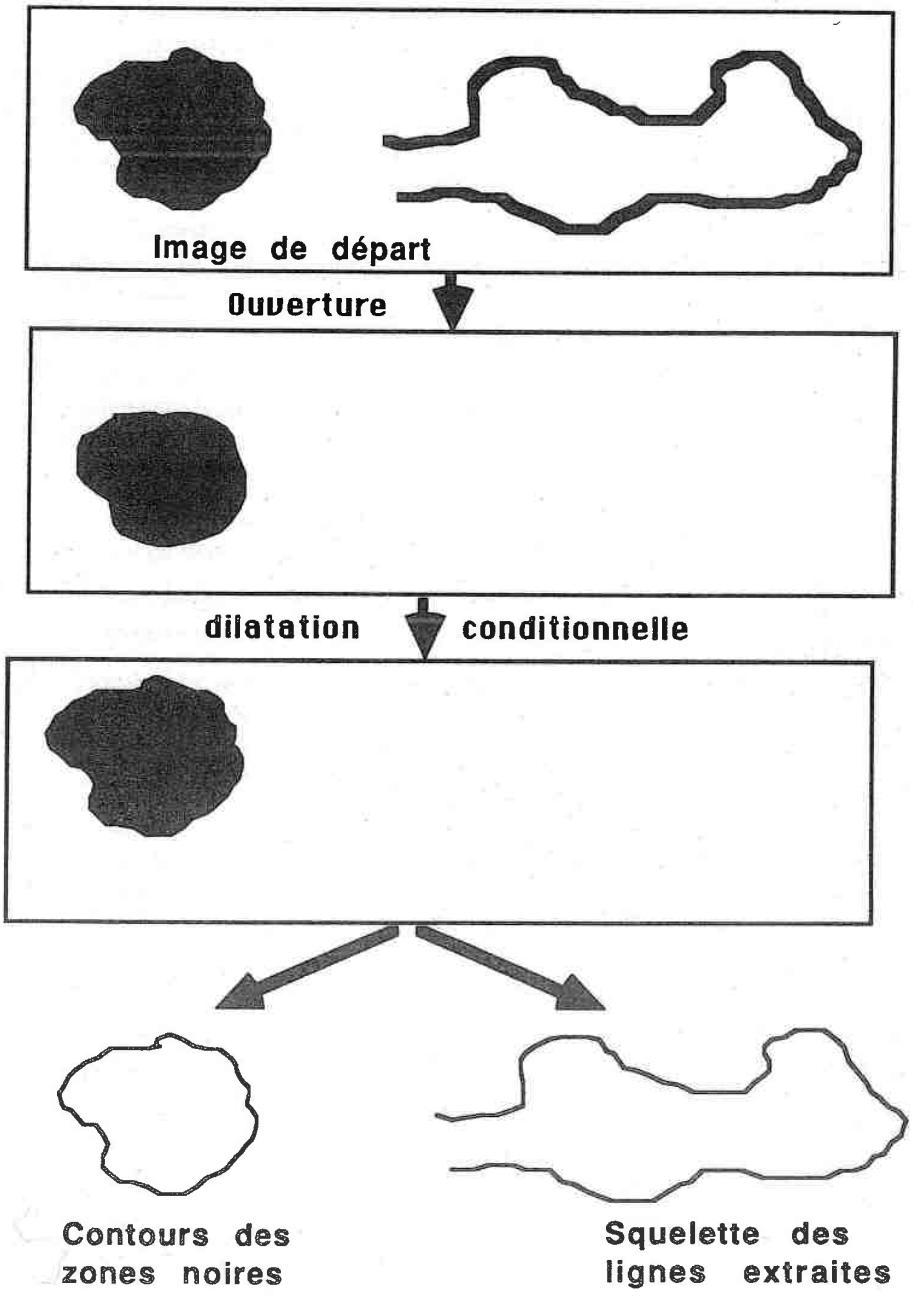


Figure 11. Extraction de lignes et contours

Les différentes étapes de la méthode proposée sont schématisées par la figure 11. Comme ce module doit s'intégrer dans un système plus vaste de reconnaissance de zones graphiques, il faut que les segments soient intégrés dans une structure de données assez riche pour permettre des analyses de plus haut niveau. Nous appelons **point particulier** un point de jonction ou un point extrémal. Nous appelons **branche** une suite de segments entre deux points particuliers. Proposons alors la démarche suivante :

- A tout moment, la vision qu'on a de l'image à traiter est purement **locale** : on voit la ligne courante, la ligne qui la précède et celle qui la suit. Par ailleurs, un certain nombre de branches sont **en cours** de suivi.
- Un point de la ligne courante peut être ajouté à une branche courante. Pour cela, nous nous servons d'une technique de marquage : au moment où nous mettons à jour un segment, nous regardons sur la ligne suivante et à droite du point courant si ce segment pourrait s'y poursuivre de manière unique ; si c'est le cas, le point correspondant est **marqué** par un pointeur vers le segment courant, comme suite potentielle de la branche.
- Pour une même branche, on décide d'arrêter le segment courant et d'en commencer un autre sur le critère de Wall et Danielsson.
- S'il n'y a plus de point suivant, la branche est arrêtée. S'il y a plusieurs points suivants potentiels, on détecte un point de jonction, et la branche courante est également arrêtée.
- A chaque point particulier auquel on démarre une ou plusieurs nouvelles branches, on associe un compteur du nombre de branches qui en partent. Quand une branche est arrêtée, le compteur associé à son point de départ est décrémenté. Quand ce compteur passe à zéro, le point en question peut être sauvegardé, ainsi que les branches associées.
- Aux points de jonction, comme nous l'avons dit, les branches aboutissantes sont arrêtées, et de nouvelles sont démarrées le cas échéant. Si la jonction se limite à l'arrêt de deux branches, sans démarrage d'une nouvelle, les deux branches sont raccordées pour n'en former qu'une ; cela implique le «retournement» des segments d'une de ces deux branches.

Cette méthode permet l'extraction d'un ensemble de segments regroupés en branches, qui elles-mêmes s'arrêtent soit sur des points extrémaux, soit sur des points de jonction. Cette structure peut être exploitée dans la suite pour trouver des primitives graphiques comme des courbes ouvertes ou fermées ; nous verrons par la suite comment de telles branches peuvent être suivies pour retrouver ces primitives.

La figure 12 présente des résultats donnés par cette approche. Elle a l'avantage de bien s'inscrire dans une suite d'opérations **locales** et faciles à implanter dans des circuits spécialisés, ce qui en fait une bonne approche pour une **machine à vectoriser** des plans techniques, par exemple.

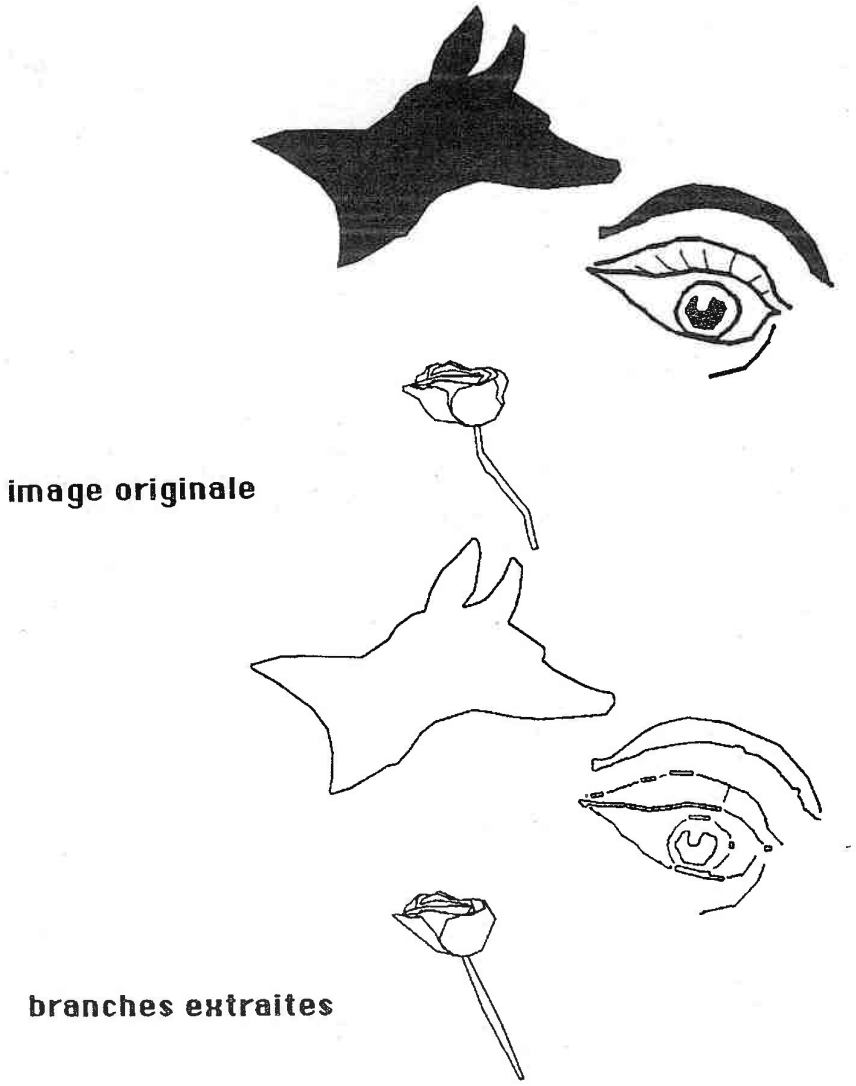


Figure 12. Extraction des branches

Toutefois, un problème majeur de toute l'approche fondée sur la squelettisation est le squelette lui-même. En effet, un squelette à souvent tendance à «baver» pour la moindre irrégularité du contour. Evidemment, il est possible **d'ébarbuler** le squelette, mais ce sera au risque de raccourcir les extrémités libres significatives. De plus, les extrémités à bord droit de lignes d'une certaine épaisseur donnent des squelettes à deux branches. Enfin, la segmentation arbitraire entre tache noire et ligne, pour nécessaire qu'elle soit, pose des problèmes quand une ligne a une épaisseur variant faiblement de part et d'autre du seuil d'épaisseur fixé. Le raccord semble dans ce cas particulièrement difficile à faire. Ces différents problèmes sont illustrés par la figure 13 ; on voit aussi certaines lignes morcelées à cause de ce genre de problèmes en figure 12.

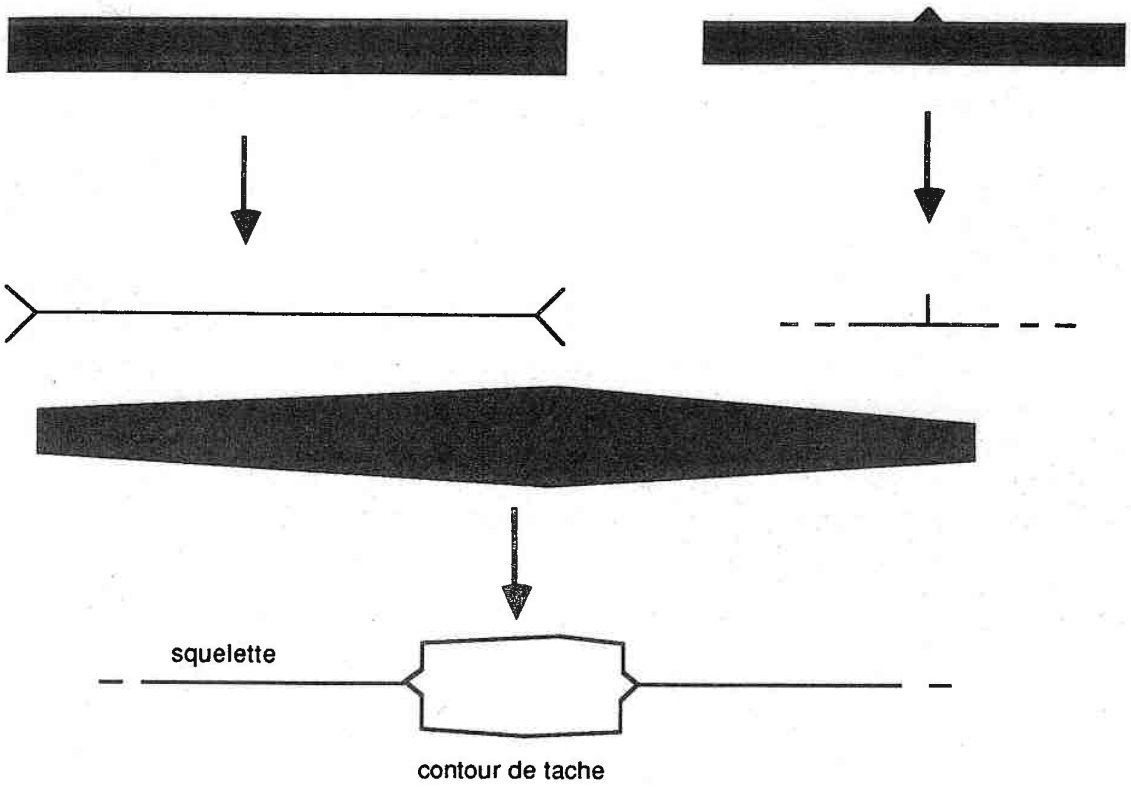


Figure 13. Problèmes de la méthode utilisant le squelette

5.3 Une approche nouvelle par les contours des composantes

Les problèmes que nous venons de citer ne sont sûrement pas insurmontables, et l'approche par le squelette n'est pas à rejeter de manière systématique. Mais dans notre cas, une autre considération de taille est venue s'ajouter : l'extraction de lignes vient naturellement s'insérer après la segmentation entre texte et graphique. Or, comme nous l'avons déjà signalé, et cela sera présenté plus longuement au chapitre suivant, la segmentation choisie dans le cadre du projet s'appuie sur l'analyse des composantes connexes. Cela implique qu'au sortir de cette segmentation, nous disposons d'un arbre de composantes connexes noires et blanches, avec leurs relations d'inclusion, de proximité, et un codage par chaînage de leurs contours.

Il est alors dommage d'abandonner toute cette information, en revenant à une image binaire, pour squelettiser et faire l'approximation polygonale, surtout en pensant à des reconnaissances de primitives graphiques de plus haut niveau, comme les zones hachurées, où nous serons obligés de retrouver ces

informations d'inclusion et de proximité. L'idée nous est alors venue **d'opérer directement sur la structure** fournie par la segmentation pour la suite des traitements. Cela impliquait d'extraire les lignes à partir des contours, par **mise en correspondance** des contours parallèles ou quasi-parallèles qui encadrent une ligne.

A notre connaissance, cette voie n'a presque pas été explorée, du moins pour la reconnaissance de graphiques. Toutefois, Cugini *et al.* [14] ont publié un article montrant des résultats encourageants. Leur approche est particulièrement originale dans le sens où ils ont un système d'analyse à base de **règles de réécriture** opérant sur un codage par chaînage du contour de l'image binaire. Ces règles leur permettent aussi bien de lisser ce contour, de distinguer les longues lignes droites, les courbes et les petits segments, que de faire des appariements de structures analogues situées l'une en face de l'autre, ce qui permet l'extraction des lignes. L'appariement ne se fait pas seulement sur un couple de segments de contours, mais sur deux ensembles de ces segments, deux segments consécutifs pouvant le cas échéant être séparés par un embranchement. Cela permet l'extraction directe de la ligne droite la plus longue, sans interruption aux intersections. La figure 14 illustre un tel appariement.

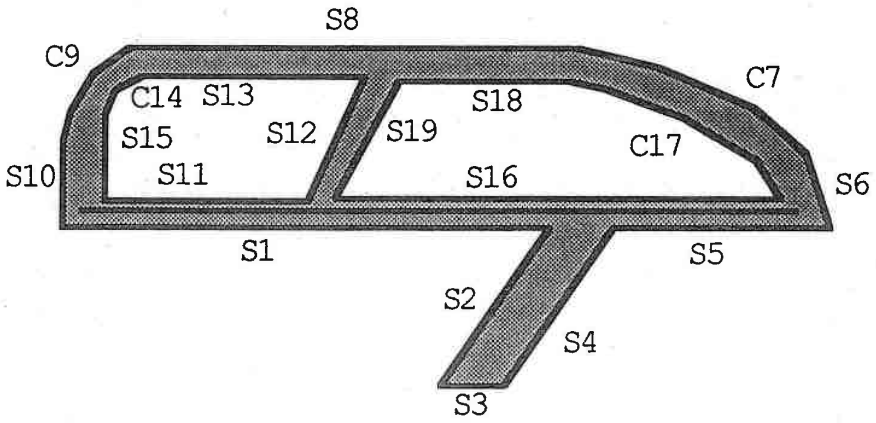


Figure 14. La méthode d'appariement de Cugini *et al.*

Les idées principales d'appariement sont intéressantes. Les résultats obtenus et présentés sont impressionnants. Toutefois, la méthode n'est pas assez générale pour s'adapter telle quelle à un document composite où on ne trouve pas seulement de beaux dessins techniques, mais aussi du graphique plus tourmenté, sans belles droites ni courbes régulières. De plus, s'il peut à première vue sembler intéressant d'extraire directement les lignes par delà les points d'intersection avec d'autres lignes, il nous a semblé préférable, dans notre cas, de donner une grande importance à ces points d'intersection, et donc de les traiter de manière spécifique. En effet, ils ont souvent une importance capitale pour caractériser les objets graphiques.

Notre approche est donc relativement proche des idées présentées ci-dessus, mais elle s'intègre dans la chaîne de traitements préalables, venant en succession de la segmentation texte / graphique [53]. Ses grandes étapes sont :

- l'approximation polygonale des contours de composantes connexes, par une méthode de suivi séquentiel, et avec le critère de Wall et Danielsson,
- l'extraction de lignes par appariements de segments, et extraction des contours quand un tel appariement n'est pas possible (cas des «taches» noires ou blanches),
- le traitement des points de jonction, c'est-à-dire le calcul de leurs coordonnées et leur rattachement aux différentes branches y aboutissant,
- le suivi des courbes pour extraction de primitives graphiques élémentaires.

Tous ces traitements sont faits sur la structure des composantes, qui s'enrichit donc à chaque nouvelle étape. Cela permet de préserver le maximum d'information pour les niveaux supérieurs. Dans le chapitre suivant, nous présentons par le détail les méthodes implantées pour réaliser ces différentes étapes.

Chapitre 6

Le module de saisie de documents dans HERODE

> ... conduire par ordre mes pensées, en commençant par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu, comme par degrés, jusques à la connaissance des plus composés ; et supposant même de l'ordre entre ceux qui ne se précédent point naturellement les uns les autres.
> Descartes

Dans ce chapitre, nous présentons le module ADE de HERODE, qui est responsable de la saisie automatisée de documents et de leur codage selon la norme ODIF. Ce travail a été mené par trois équipes appartenant à des institutions différentes et géographiquement séparées. Il est évident que nous insisterons bien plus sur notre propre contribution, qui porte sur la reconnaissance des zones graphiques. Toutefois, il nous semble difficile de ne pas replacer cette contribution dans son contexte d'ensemble, d'autant plus qu'une partie du travail commun a justement été **l'intégration** des programmes écrits par chacun pour en faire un tout cohérent. De plus, la nature même de notre sujet nous obligeait à suivre de près l'élaboration de certains autres modules, et en particulier de la segmentation entre texte et graphique, puisque l'implantation choisie avait une incidence directe sur les méthodes que nous devions choisir. A cet égard, les longues séances de discussion et d'élaboration commune, avec l'équipe de TITN en particulier, ont porté des fruits dont nous sommes tous satisfaits, du moins je l'espère.

6.1 Acquisition, prétraitements et segmentation photo / binaire

Le document est saisi à l'aide d'une caméra CCD ayant une résolution de 200 points par pouce, ce qui correspond à 8 points par millimètre, ou aussi 4 millions de points pour une page A4. L'image est numérisée sur 8 bits, ce qui donne 256 niveaux de gris potentiels pour chaque *pixel*. Un premier prétraitement consiste à améliorer la dynamique de l'image et à corriger les défauts d'éclairage ; de simples manipulations d'histogramme suffisent pour cela.

Il faut ensuite redresser l'image qui a pu être saisie de travers. Nous avons déjà cité en grande partie la méthode développée par Postl dans le cadre de ce projet [42]. Elle consiste à passer dans le domaine fréquentiel par une transformée de Fourier et à trouver la direction donnant une valeur maximale quand on intègre le module de la transformée de Fourier sur un vecteur de cette direction. L'angle que cette direction fait avec l'horizontale est l'angle d'inclinaison à la saisie ; l'image peut ainsi être redressée par rééchantillonnage.

L'étape suivante est alors de séparer l'information binaire de l'information multi-niveaux. La méthode, mise en oeuvre par une autre équipe du projet, commence par délimiter des zones à texture homogène. Pour cela, on calcule l'histogramme de l'image et on cherche tous ses maxima locaux significatifs. Ces différents maxima définissent plusieurs intervalles ; un seuillage où on ne retient que les points compris dans un de ces intervalles donne un sous-ensemble de points de l'image. L'analyse de la répartition des points de ce sous-ensemble, couplée à une analyse de Fourier des régions connexes du sous-ensemble, mène à une décision sur le type de contenu de ces régions. Le résultat final est un masque binaire indiquant les parties de l'image d'origine qui doivent être considérées comme à multi-niveaux, et par conséquent codées en tant que telles. La figure 15, tirée d'un article commun mais due à Mr. Postl, illustre les étapes de cette segmentation.

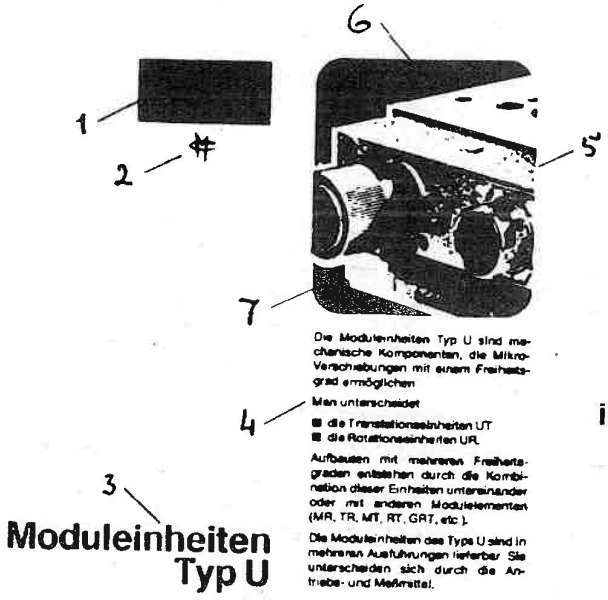
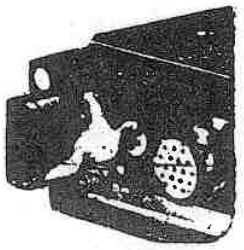


image originale



zones photographiques

masque des zones photographiques

Figure 15. Segmentation binaire / multi-niveaux

6.2 Binarisation et segmentation texte / graphique

A partir de maintenant, nous ne nous intéressons plus qu'aux zones où l'information est binaire. L'image reste cependant à niveaux de gris, et le premier travail est par conséquent la binarisation. Comme nous l'avons vu dans le chapitre sur le prétraitement, la binarisation doit être adaptative car on ne peut pas garantir que l'image de départ est toujours d'excellente qualité.

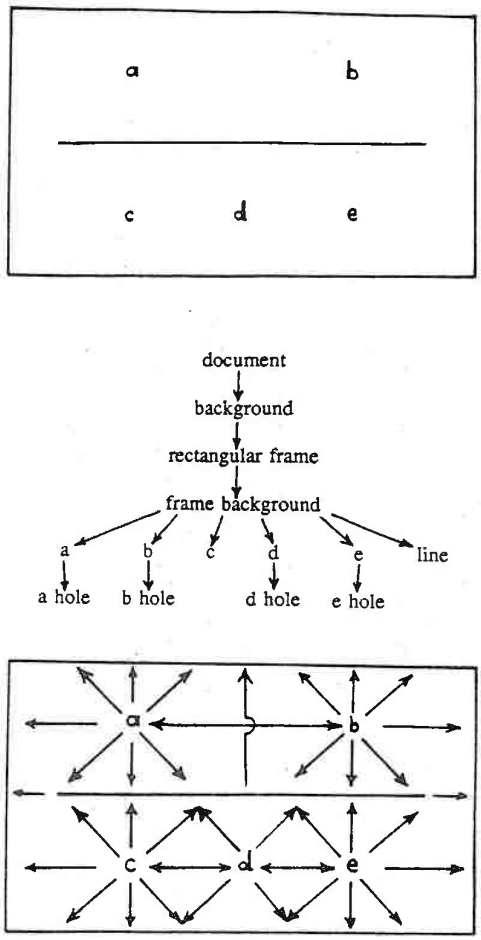
La méthode mise en oeuvre reprend quelques idées développées par White et Rohrer [59], idées que nous avons déjà abordées dans un précédent chapitre. En effet, on commence par détecter les contours de l'image grâce à un pseudo-laplacien, puis on remplit ces contours par du noir, obtenant ainsi une image binaire. Cette image est ensuite filtrée : dans les directions horizontale et verticale, on élimine les petites plages noires et blanches jugées insignifiantes. Une plage est jugée insignifiante si elle est de longueur inférieure à n (où n est un facteur dépendant de la résolution) et si, dans la direction perpendiculaire, elle est bordée par deux plages de couleur opposée et de longueur supérieure à n.

En nous référant aux propriétés idéales définies au chapitre 2, ce filtrage lisse les contours, préserve l'information d'épaisseur 1, préserve la connexité. De plus, il est idempotent et a été avéré robuste.

Il faut ensuite effectuer la segmentation entre graphique et texte. Le choix fait pour cela est de s'appuyer sur les composantes connexes. Une structure arborescente de composantes connexes noires et blanches est créée ; la relation hiérarchique dans cet arbre est celle **d'inclusion**, que nous avions déjà illustrée par la figure 5. A chaque composante est associé le codage de son contour, par une suite de portions de contours codées par un codage dérivé de celui de Cederberg ; ainsi la structure peut être créée en un seul parcours de l'image. Les différentes portions d'un même contour sont chaînées en avant et en arrière, pour permettre un parcours continu du contour sur la structure ainsi créée.

La structure est complétée par des informations internes à chaque composante, comme la position, le rectangle englobant, l'excentricité, la densité de points, le centre de gravité, la surface etc. ainsi que par des liens de proximité entre composantes d'un même niveau hiérarchique. Deux composantes sont en relation de proximité horizontale ou verticale si on peut tracer une ligne de cette direction entre elles qui les joigne sans couper une autre composante de la même couleur. Par ailleurs, chaque composante est en relation de proximité avec celle qui en est la plus proche. A ce niveau, on inclut les contours du masque photographique fourni par la segmentation multi-niveaux / binaire, dans la mesure où il peut avoir une influence sur ces relations topologiques entre composantes.

La figure 16, tirée de [33] et due à Eric Meynieux qui est le principal concepteur de la méthode de segmentation mentionnée ici, illustre les relations de proximité sur une image binaire simple.



Up : Original image
Middle : Hierarchical structure
Down : Proximity relationships

Figure 16. Relations de proximité

La structure ainsi créée est alors analysée pour permettre une segmentation en plusieurs classes. Pour commencer, les composantes ou sous-arbres considérés comme trop complexes pour avoir une chance d'être correctement reconnus sont rejetés dans une classe *rebut* ; ils seront représentés par un codage de type fac-similé dans le résultat final. On retrouve dans cette classe, entre autres, l'écriture manuscrite, les signatures, les dessins compliqués. Le critère de complexité est évalué essentiellement à partir des informations internes de la composante.

On applique ensuite un ensemble de **règles de décision** aux composantes restantes pour déterminer leur type. Contrairement à beaucoup d'autres systèmes, aucune règle sur la taille de la composante n'est utilisée ; on ne fixe donc pas une taille maximale au-delà de laquelle on ne trouvera plus aucun caractère. Les règles basées sur l'information interne sont par conséquent limitées. On décide toutefois qu'une composante ayant une trop forte excentricité ne peut *a priori* être un caractère.

L'essentiel des règles porte sur les relations topologiques de la structure. Le nombre de niveaux d'inclusion en-dessous du niveau courant, par exemple, permet parfois de rejeter l'hypothèse *texte*. Les relations de proximité donnent aussi de nombreuses indications, et permettent par ailleurs de regrouper les composantes en blocs de même contenu. Un tel regroupement entraîne d'ailleurs souvent une remise en cause de certains choix antérieurs ; le système de règles permet bien sûr de tels retours en arrière.

Le résultat final est une segmentation fiable en texte d'une part et graphique de l'autre. La figure 17 montre un résultat typique de cette segmentation. On trouvera aussi des exemples intéressants en annexe, dans la présentation des résultats. Le texte est regroupé si possible en lignes et paragraphes, mais des caractères isolés n'en sont pas pour autant perdus. Il est envisageable à cette étape de soumettre le texte à un système de lecture optique de caractères. Toutefois, pour notre part, nous ne nous occuperons plus à partir d'ici que des parties graphiques.

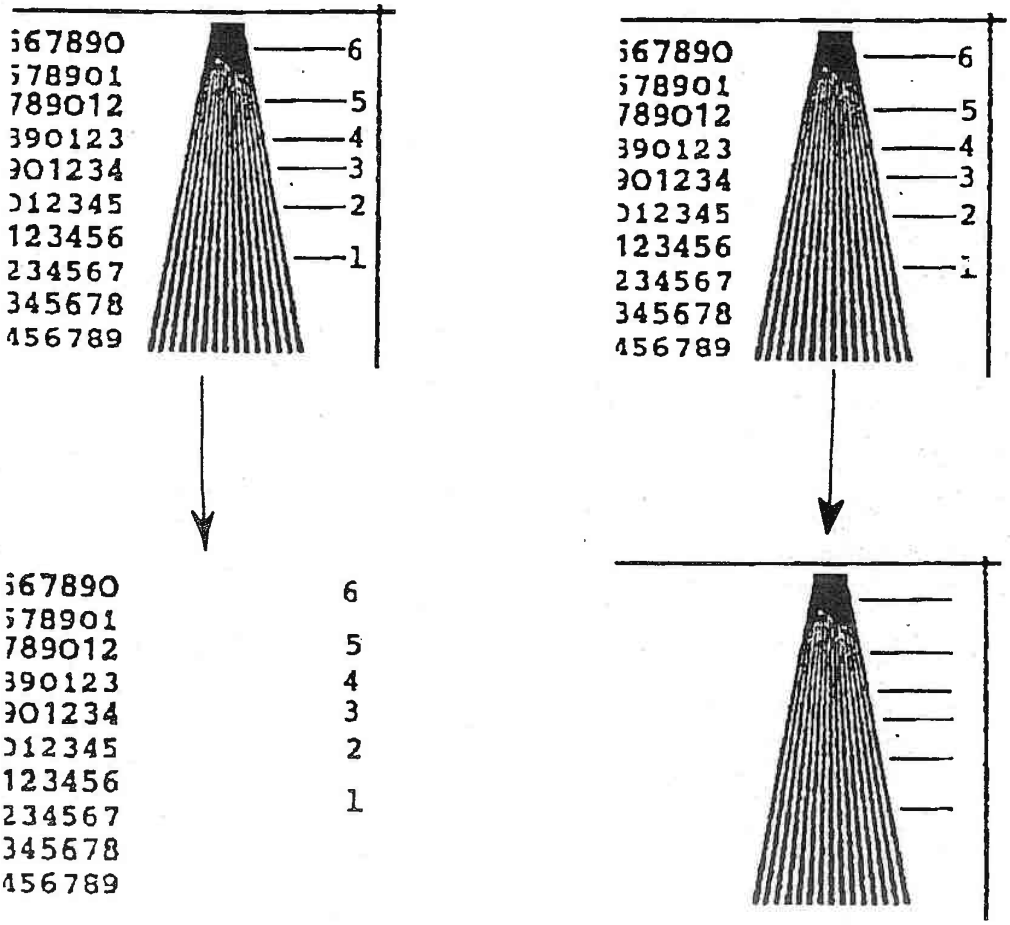


Figure 17. Segmentation texte / graphique

6.3 Reconnaissance des primitives graphiques

Nous entrons ici dans le vif de notre sujet, puisque les méthodes présentées ci-après ont été développées par nous, la reconnaissance du graphique étant notre responsabilité dans le cadre de ce projet. L'existence de la structure présentée précédemment, issue de la segmentation entre texte et graphique, nous a incités à mettre au point une méthode opérant directement sur cette structure, pour ne pas perdre les informations

structurelles et topologiques. Cela implique bien sûr l'adaptation des méthodes de traitement employées à cette structure.

6.3.1 Approximation polygonale

Nous disposons, pour chaque composante connexe noire ou blanche, d'un codage de ses contours. Le premier traitement que nous appliquons est l'approximation polygonale de ces contours. Cela nous permet par la suite de travailler sur une suite de segments, ce qui est une première étape vers l'extraction des structures linéaires. Cette approximation, comme tous les traitements subséquents, n'est effectuée que sur les composantes reconnues comme de type graphique par la segmentation. Il faut noter pour la bonne compréhension de ce qui suit que si une composante connexe noire est reconnue de type graphique, ses filles blanches le sont aussi. Toutefois, le fait qu'une composante noire à un niveau donné soit graphique n'implique nullement que tout l'arbre en-dessous le soit : on peut très bien trouver du texte dans un cadre par exemple.

Nous avons choisi d'implanter la méthode d'approximation polygonale de Wall et Danielsson avec préservation des points anguleux. L'algorithme a déjà été donné dans le chapitre sur l'approximation polygonale. Le parcours du contour se fait dans le sens des aiguilles d'une montre pour les composantes noires et dans le sens inverse pour les blanches. Il faut noter que par leur nature même, les contours sont des courbes fermées.

En ce qui concerne le sens de parcours, remarquons tout de suite que le repère choisi est le repère usuel des programmes graphiques sur ordinateur, où l'origine est en haut à gauche, l'axe des x va de gauche à droite et l'axe des y de haut en bas. Par conséquent, le sens des aiguilles d'une montre correspond dans ce repère au sens trigonométrique direct.

A l'issue de l'approximation polygonale, on a ainsi associé à chaque composante une liste doublement chaînée de segments. Le lien de chaînage direct est le sens trigonométrique direct pour les composantes noires et le sens indirect pour les blanches ; cela nous permet de savoir qu'à tout moment, quand on parcourt la liste de segments, le **noir** est sur la droite et le **blanc** sur la gauche. A chaque segment sont associées un certain nombre d'informations :

- les coordonnées de son point origine,
- sa longueur (ou plutôt le carré de sa longueur pour travailler avec des entiers),
- une mesure de l'aire algébrique comprise entre le segment et la courbe qu'il approche ; cette mesure est celle fournie par la méthode de Wall et Danielsson,

- des pointeurs sur le segment précédent et le segment suivant,
- la direction générale du segment, codée sur un octet et reprenant le codage de Freeman ; cette direction générale est uniquement un moyen d'éliminer rapidement, en phase d'appariement, les segments ayant une direction trop éloignée de celle cherchée pour qu'il soit nécessaire de poursuivre les essais d'appariement.

D'autres paramètres sont aussi associés au segment, mais ils n'interviennent qu'à une étape postérieure. La figure 18 illustre ces différentes informations associées aux segments.

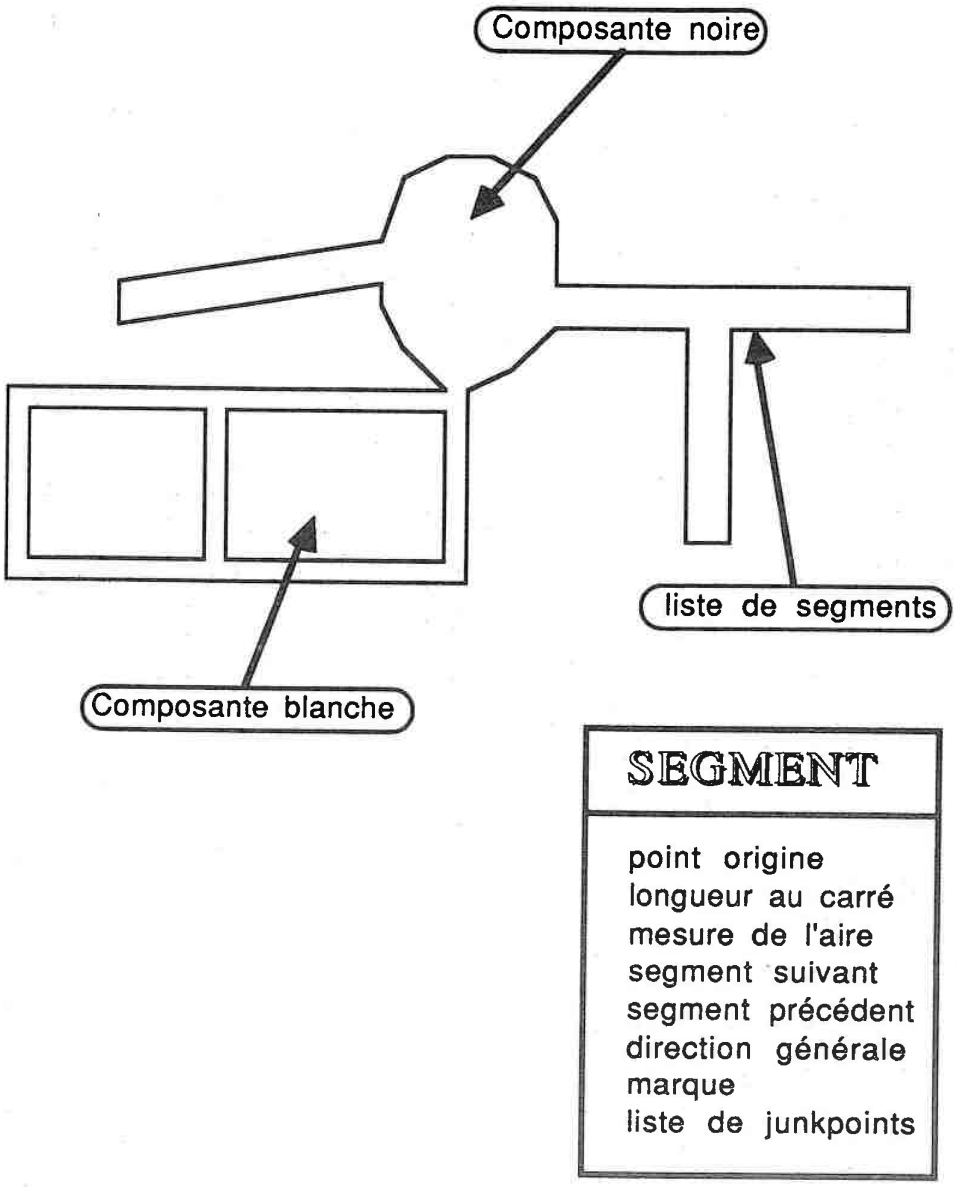


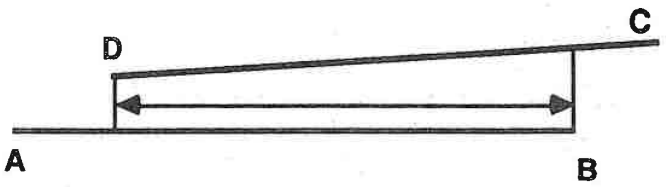
Figure 18. Approximation polygonale sur la structure

6.3.2 Extraction des lignes

Il s'agit maintenant d'extraire les lignes quand il y en a, par appariement des segments de droite.

Contrairement à Cugini qui cherche à apparier deux ensembles de segments, nous avons choisi de n'apparier qu'un segment avec un autre. Pour que deux segments **seg1** et **seg2** soient appariés, les conditions suivantes doivent être remplies :

- Le noir étant toujours à droite de tout segment quand on suit la liste de segments dans le sens de chaînage direct, on ne peut trouver une ligne qu'entre deux segments de directions opposées. La direction générale codée sur un caractère nous permet ainsi d'éliminer à moindres frais un grand nombre d'hypothèses d'appariements impossibles.
- Les deux segments doivent être en face l'un de l'autre, la partie sur laquelle ils le sont étant de longueur non négligeable. Nous déterminons ce critère par un raisonnement trigonométrique simple illustré par la figure 19.



cos(AB,AC) > 0 et cos(BA,BD) > 0
et longueur commune > seuil

Figure 19. Les deux segments doivent être en face l'un de l'autre

- Les deux segments étant en face l'un de l'autre, il faut encore qu'ils soient du bon côté l'un de l'autre, c'est-à-dire qu'il y ait bien du noir entre les deux. C'est ce critère qui en pratique s'est avéré être le plus délicat à appliquer. En effet, l'approximation polygonale engendre des erreurs dans la position des segments l'un par rapport à l'autre. Ainsi, on peut trouver deux segments qui se coupent ou même qui sont inversés dans leur position l'un vis à vis de l'autre. Nous avons résolu ce problème en plusieurs étapes, illustrées par la figure 20 :

- si les deux segments se coupent, l'appariement est possible ;
- sinon, si les deux segments sont bien du bon côté l'un de l'autre, l'appariement est aussi possible ;
- enfin, si ce n'est pas le cas, nous considérons pour chacun des deux segments un point virtuel, sommet d'un triangle dont la base est le segment et l'aire algébrique celle associée au segment par l'approximation polygonale. Ces triangles virtuels sont en fait une estimation grossière de la courbe originale. L'appariement est alors valide si les deux points virtuels sont bien du bon côté l'un de l'autre ;
- dans les autres cas, l'appariement est rejeté.

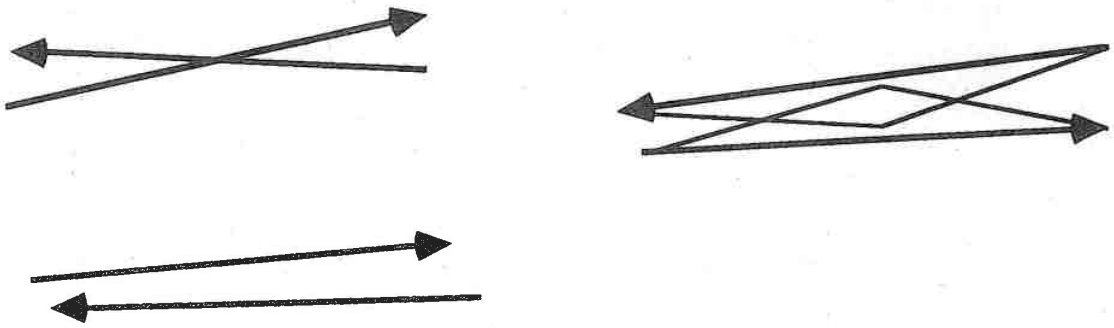


Figure 20. Appariements possibles

- La distance qui les sépare doit être inférieure à un seuil. Ce seuil est un paramètre du traitement, dépendant du type de figure qu'on risque de rencontrer et de la résolution, puisque la distance ici est un nombre de *pixels*.

Comme l'indique la figure 21, il y a trois types d'appariements : deux segments noirs de la même composante, un segment noir et un segment blanc d'une composante fille, deux segments blancs de composantes soeurs.

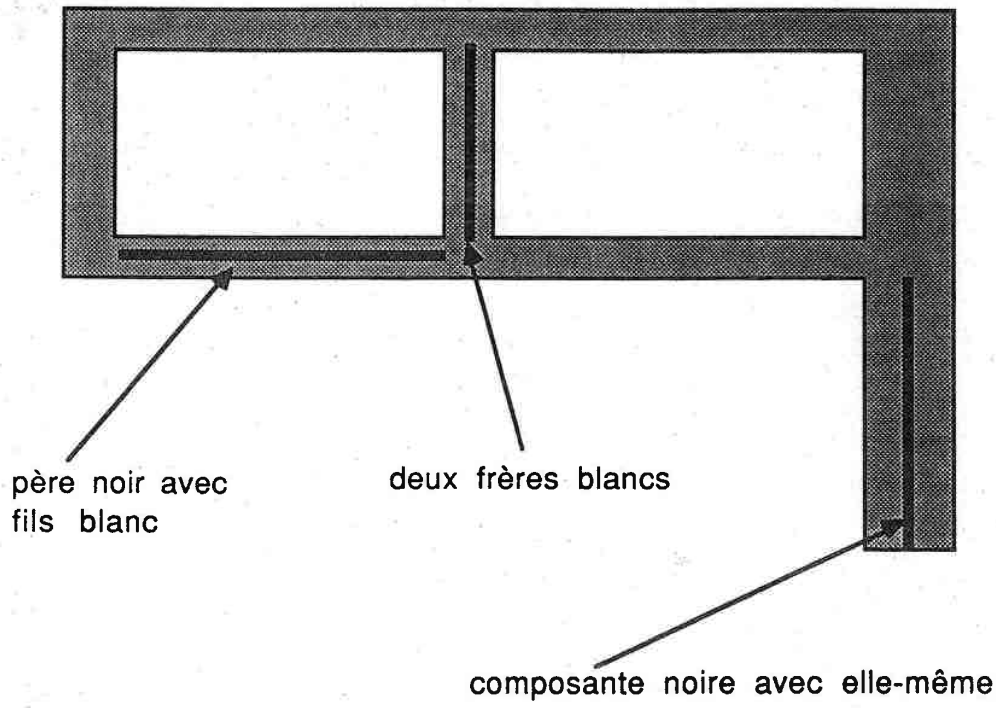


Figure 21. Les différents types d'appariement

Quand un appariement a été trouvé, on suit la ligne aussi loin que possible, ce qui donne ce que nous avons appelé une **branche**. On arrête la branche quand l'appariement n'est plus possible, ou quand on arrive sur des segments déjà appariés. On l'arrête aussi si l'épaisseur de ligne au cours du suivi varie plus qu'un seuil toléré par rapport à l'épaisseur originale trouvée. Une branche est donc constituée d'une liste de segments de droite. Quand aucun appariement n'est possible, on en déduit qu'on est sur une grosse zone noire ou blanche ; dans ce cas, ce sont les contours de cette zone qui sont suivis tant qu'un appariement reste impossible. A chaque segment apparié, on associe également un pointeur sur la ligne créée par l'appariement. Quand une partie non négligeable d'un des segments appariés dépasse la zone où ils sont en face l'un de l'autre, on découpe le segment en deux pour permettre un autre appariement sur la partie restante.

L'algorithme général d'extraction des lignes sur une composante s'écrit alors :

```
tantque tous les segments n'ont pas été marqués faire

    chercher un segment non marqué sur la composante ou
    sur une de ses filles blanches ;

    si il existe un appariement possible pour le segment alors
       démarrer une nouvelle branche de type ligne ;
       tantque l'appariement reste possible et
                l'épaisseur de ligne ne varie pas trop et
                au moins un des segments est non marqué faire
           prolonger la branche par suivi double ;
           découper éventuellement un segment qui dépasse trop ;
           marquer les deux segments appariés ;
       ftantque
       arrêter la branche de type ligne ;
    sinon
       démarrer une nouvelle branche de type contour ;
       tantque l'appariement reste impossible faire
           prolonger la branche par suivi simple du lien de
           chaînage entre segments ;
           marquer le segment pris en compte ;
       ftantque
       arrêter la branche de type contour ;
    fsi
ftantque
```

6.3.3 Traitement des points de jonction

Pendant l'extraction des lignes, on détecte aussi les points de jonction éventuels. En effet, quand on arrête une branche, il y a soit un point de jonction avec une autre branche, soit un point extrémal. Quand on démarre et quand on arrête une branche, le chaînage sur les segments permet de déterminer quelles autres branches sont en jonction avec la branche courante. Toutefois, toutes ces branches n'ayant pas forcément été créées encore, il nous faut passer par une structure intermédiaire, appelée *junkpoint*, ou faux point de jonction, pour pouvoir gérer les points de jonction au fur et à mesure du parcours. Chaque fois qu'une branche est commencée ou terminée, un tel *junkpoint* est créé. A chaque vrai point de jonction est associée une liste de *junkpoints*. On associe aussi à chaque segment une paire de pointeurs sur les *junkpoints* associés respectivement à son origine et à son extrémité. Nous donnons ci-dessous l'algorithme dans le cas d'une branche qui est arrêtée ; il est clair que le début d'une branche est similaire. La branche est arrêtée sur l'appariement de `seg1` et `seg2` ; comme ils sont suivis dans des directions opposées, nous supposerons que l'extrémité de la branche correspond à l'extrémité de `seg1` et à l'origine de `seg2`. Dans certains cas, un nouveau *junkpoint* créé fait le lien entre deux points de jonction existants, ce qui entraîne leur fusion :

```
créer un nouveau junkpoint newjk;

nsegl = successeur de segl dans la liste des segments;
nseg2 = prédécesseur de seg2 dans la liste des segments;

jkl = junkpoint éventuel associé à l'origine de nsegl;
jk2 = junkpoint éventuel associé à l'extrémité de nseg2;

si jkl existe alors
    newjk est associé à la même jonction que jkl;
    si jk2 existe alors
       si jkl et jk2 sont associés à deux jonctions différentes alors
           fusionner ces deux jonctions car l'arrêt de la branche
           courante nous indique que c'est la même jonction;
sinonsi jk2 existe alors
    newjk est associé à la même jonction que jk2;
sinon
    créer une nouvelle jonction newjp à laquelle sera associé newjk;
fsi

associer newjk à l'extrémité de segl;
associer newjk à l'origine de seg2;
```

Les notions de *junkpoint* et de point de jonction sont illustrées par la figure 22 :

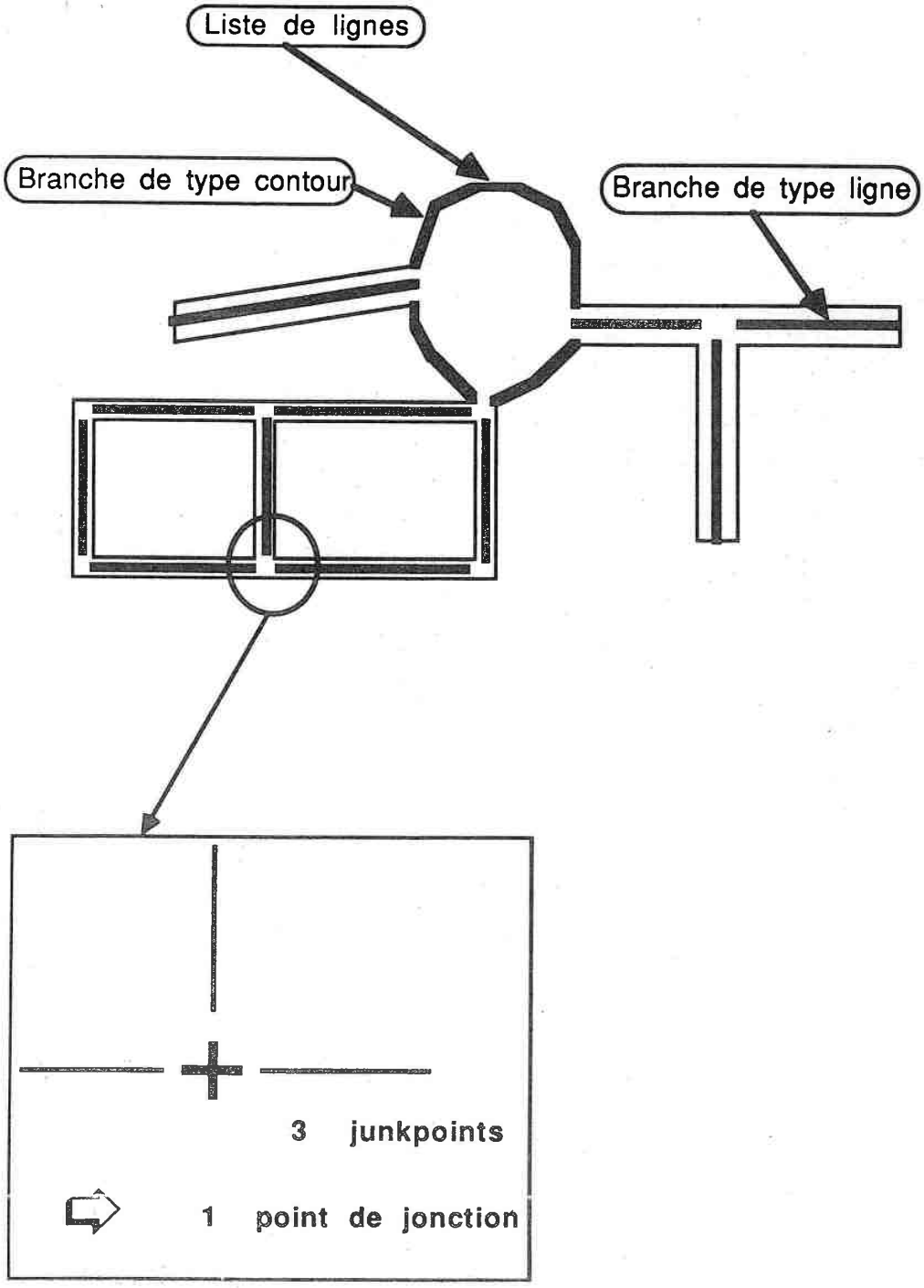


Figure 22. Points de jonctions et *junkpoints*

C'est seulement quand toutes les lignes ont été extraites que nous calculons les coordonnées de tous les points de jonction trouvés, ce qui nous permet de connecter les différentes branches. Pour éviter dans toute la mesure du possible de déformer les courbes par ce calcul, nous appliquons un certain nombre de règles suivant les différents cas de figure possibles :

- s'il y a une seule courbe, le point de jonction (qui est en fait un point extrémal dans ce cas) est évident,
- s'il y a 3 courbes ou plus, on prend le barycentre des différents *junkpoints*,
- s'il y a 2 courbes, on distingue le cas où l'angle entre les deux est plutôt plat de celui où il est plutôt droit. Dans le cas d'un angle quasi-droit (plus de 60 degrés), le point de jonction sera l'intersection des deux derniers segments quand on les prolonge. Dans l'autre cas, on prendra le point milieu des deux *junkpoints*.

La figure 23 illustre ces différents cas. Evidemment, les branches sont ajustées en conséquence au niveau de ces points de jonction, pour que l'ensemble de la structure reste cohérent.

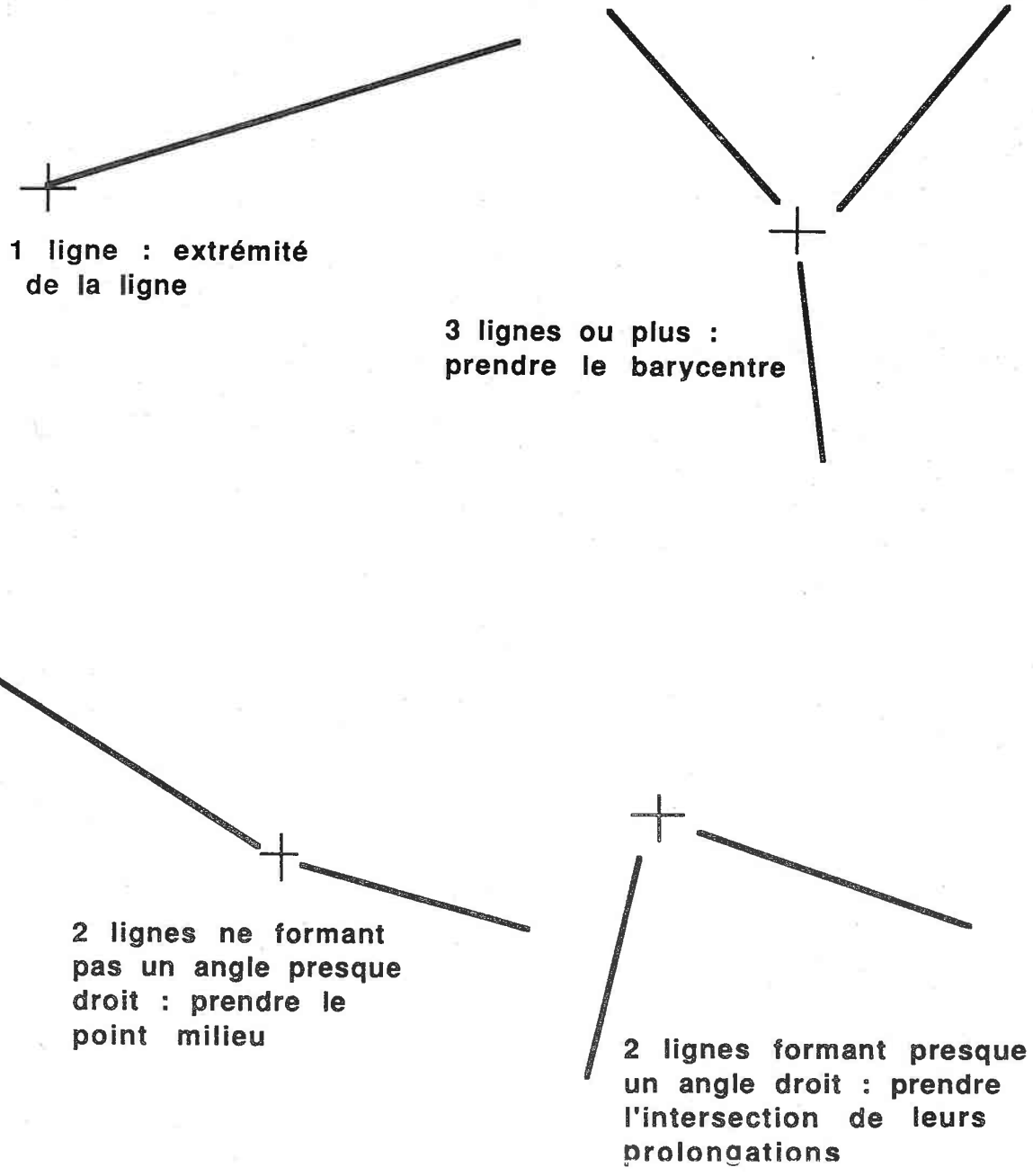


Figure 23. Calcul des points de jonctions

6.3.4 Reconnaissance de primitives élémentaires

Le résultat des différents traitements présentés jusqu'ici est un ensemble de branches se raccordant en des points de jonction. Concrètement, une liste de branches et une liste de points de jonction sont associées à chaque composante **noire** reconnue comme graphique. Il s'agit maintenant de franchir une étape supplémentaire en regroupant ces branches en **courbes** ouvertes ou fermées. Ces courbes seront nos primitives graphiques élémentaires.

Il faut toutefois que le suivi de courbes donne des résultats **fiables** et surtout **conformes à l'interprétation humaine** de la figure géométrique. En travaillant toujours sur la même structure de composantes connexes, qui s'est déjà enrichie considérablement au fil des traitements, nous avons mis au point un module d'extraction de courbes donnant des résultats satisfaisants.

Le concept de base maintenant est donc une **courbe**. Elle a les caractéristiques suivantes :

- l'épaisseur de trait,
- le type de courbe (ligne noire, contour noir ou contour blanc),
- le fait que ce soit une courbe ouverte ou fermée,
- une liste de segments de courbe.

Un **segment de courbe** est caractérisé par :

- ses points origine et extrémité,
- deux liens de chaînage pour que les segments soient dans une liste doublement chaînée,
- une liste de jonctions de courbes associées au segment,
- un pointeur vers la courbe d'appartenance.

Une **jonction de courbes** est un point où deux courbes se rencontrent. Contrairement à ce qui se passe pour les branches extraites ci-dessus, une courbe peut en rencontrer une autre non seulement à ses extrémités, mais aussi n'importe où ailleurs.

Le principe de la méthode de reconnaissance de courbes ouvertes et fermées est le suivi des branches par-delà les points de jonction. Pour que les résultats soient satisfaisants, ce suivi doit faire les choix les plus judicieux sur les branches à emprunter quand il y a plusieurs possibilités. Par **choix judicieux**, nous entendons un choix qui correspond dans toute la mesure du possible à ce que voit l'homme. Ainsi, en présence d'un rectangle quadrillé, on s'attend à trouver le rectangle le plus grand d'un côté et un ensemble de courbes ouvertes de l'autre, et non une juxtaposition de petits rectangles.

Pour obtenir de tels résultats, nous avons appliqué plusieurs principes. Le premier est de toujours partir d'un point de jonction présentant le moins de chances de se tromper. C'est pourquoi on privilégie les points de jonction à partir desquels ne part qu'une branche non traitée, ainsi que ceux qui permettent une continuation de la courbe dans le même direction. Le second principe est de ne continuer le suivi de courbe que s'il n'y a pas d'ambiguïté flagrante. Ainsi, quand on aboutit à un point de jonction à partir duquel plusieurs branches peuvent être empruntées, on s'arrête à moins qu'une de ces branches est dans la prolongation de la courbe.

A l'usage, ces heuristiques ont été avérées satisfaisantes. En particulier, les objets graphiques les plus englobants sont bien reconnus. Nous présentons dans ce qui suit les principales étapes de la méthode d'extraction de courbes pour une composante connexe donnée :

- Tant que toutes les branches n'ont pas été suivies, choisir une nouvelle branche et la suivre. Le choix de la branche sur laquelle démarrer une nouvelle courbe est fondé sur les points de jonction trouvés au cours de l'extraction de lignes. On cherche en premier un point de jonction à partir duquel ne part qu'une branche non traitée. Si un tel point de jonction existe, c'est à partir de lui que la nouvelle courbe commencera. Sinon, on en cherche un à partir duquel ne partent que deux branches non traitées, car il ne peut alors y avoir d'ambiguïté. Si un tel point est trouvé, on suit successivement les deux branches et on fusionne ensuite les deux courbes ainsi créées. Si on n'a toujours pas trouvé de point de jonction, on en cherche un sur lequel deux des branches non traitées ont la même direction et peuvent par conséquent se prolonger. Enfin, si toutes ces recherches échouent, on est réduit à choisir une branche quelconque sur laquelle commencer la courbe. L'algorithme de recherche d'une branche de départ s'écrit :

```
trouvé = faux ;
pjonc = premier point de jonction associé à la composante ;
trouvé2 = faux ;
trouvé3 = faux ;

tantque non trouvé et pjonc existe faire
    si un seul junkpoint non marqué est associé à pjonc alors
        branche1 = branche associée à ce junkpoint ;
        trouvé = vrai ;
    sinon
        si non trouvé2 alors
            si 2 junkpoints non marqués sont associés à pjonc alors
                branche1 = 1ère branche associée à ce junkpoint ;
                branche2 = 2ème branche associée à ce junkpoint ;
                trouvé2 = vrai ;
            sinon
                si non trouvé3 alors
                    si 2 branches partant de pjonc ont
                       la même direction alors
                        branche1 = 1ère branche ;
                        branche2 = 2ème branche ;
                        trouvé3 = vrai ;
                    sinon
                        branche1 = une branche au hasard
                                   partant de pjonc ;
                    fsi
                fsi
            fsi
    fsi
    pjonc = point de jonction suivant pour la composante ;
ftantque

si trouvé alors
    partir sur branche1 ;
sinon
    si trouvé2 ou trouvé3 alors
        partir sur branche1 et sur branche2 ;
    sinon
        partir sur branche1 ;
    fsi
fsi
```

- Le suivi proprement dit revient à chaîner plusieurs branches en une courbe unique. Chaque branche suivie est évidemment marquée pour ne plus être considérée par la suite. A la fin de chaque branche, on aboutit à un point de jonction. S'il n'y a plus de branche non marquée partant de ce point, soit parce qu'on est sur un point extrémal, soit parce que toutes les branches sont

déjà marquées, la courbe est arrêtée. Sinon, s'il n'y a qu'une branche non marquée, c'est celle-là qui est prise. S'il y en a plusieurs, on cherche une branche qui prolonge la courbe par-delà le point de jonction. Si une telle branche n'existe pas, plutôt que d'en choisir une au hasard et de risquer une mauvaise interprétation, nous arrêtons la courbe. L'algorithme de suivi de courbe est donné ci-dessous :

```
arrêt = faux ;
cbranche = branche de départ ;

tantque non arrêt faire
    ajouter cbranche à la courbe ;
    marquer cbranche ;
    pjonc = point de jonction terminant cbranche ;
    nb = nombre de branches non marquées partant de pjonc ;

    si nb == 0 alors
        arrêt = vrai ;
    sinon
        si nb == 1 alors
            cbranche = branche non marquée partant de pjonc ;
        sinon
            si il existe pbranche partant de pjonc et
               prolongeant la courbe courante alors
                cbranche = pbranche ;
            sinon
                arrêt = vrai ;
            fsi
        fsi
    fsi
ftantque
```

- Quand une courbe est poursuivie par-delà un point de jonction, le segment de courbe courant peut très bien rester le même, si les deux branches sont dans le prolongement l'une de l'autre. C'est aussi dans ce cas que la jonction de courbe se trouve être au milieu du segment de courbe. De manière générale, les jonctions de courbe sont juste détectées au moment de la création de courbes ; leur coordonnées ne seront calculées qu'en dernier lieu, ce qui laisse de la souplesse pour le redressement.
- Au moment où la courbe est arrêtée, elle est considérée comme fermée si elle est revenue à son point de départ.

Nous avons parlé de **redressement**. En effet, de nombreux objets géométriques contiennent des lignes horizontales et verticales. A cause des erreurs introduites à toutes les étapes du traitement, ces lignes ne sont plus toujours parfaitement droites quand on en vient au suivi de courbes. Nous avons donc implanté un module de redressement de courbes, optionnel pour l'utilisateur puisque son activation est un paramètre du système. Passant sur la structure après extraction des courbes, il redresse les segments de courbe faisant un angle suffisamment petit avec l'horizontale ou la verticale. Au cours de ce redressement, la répercussion sur

la structure est réduite à un strict minimum : on ne modifie que les segments de courbe dont une extrémité correspond à l'extrémité modifiée du segment redressé ; ce sont donc les jonctions bout-à-bout. Les courbes coupant le segment redressé et celles qu'il coupe ne sont pas affectées dans un premier temps ; ce n'est qu'à la fin que toutes les coordonnées des points de jonction sont recalculées. L'algorithme de redressement d'une courbe `curv` est le suivant :

```
pour chaque segment seg de curv faire
    si seg forme un angle faible avec l'horizontale alors
       redresser l'extrémité de seg pour qu'il soit horizontal ;
       redresse = vrai ;
    sinon
       si seg forme un angle faible avec la verticale alors
           redresser l'extrémité de seg pour qu'il soit vertical ;
           redresse = vrai ;
       sinon
           redresse = faux ;
       fsi
    fsi

    si redresse alors
       si l'extrémité du segment redressé correspond à un point
          de jonction alors
           corriger les segments de courbes ayant leur origine ou
           leur extrémité à ce point de jonction ;
       fsi
    fsi
fpour
```

Pour terminer, que le redressement ait été activé ou non, un dernier passage sur la structure calcule les coordonnées des points de jonction. La méthode choisie permet d'éviter que cette étape altère à nouveau les effets du redressement ; les liaisons d'extrémité à extrémité ont déjà été prises en compte, les jonctions où un segment s'arrête sur un autre segment (jonctions en T) sont calculées par simple ajustement de l'extrémité du premier segment, et pour les intersections de deux segments, il suffit de calculer les coordonnées du point d'intersection.

On a donc associé à chaque composante connexe noire reconnue comme graphique un ensemble de courbes ouvertes et fermées. Ce sont des primitives graphiques élémentaires qui peuvent déjà être codées telles quelles selon la norme CGM[7], qui a été choisie dans ODIF pour le codage du graphique. Nous présentons les principales primitives graphiques de CGM en annexe.

La plupart des systèmes de reconnaissance de graphique fournissent une liste de segments ou éventuellement

---

7 CGM Computer Graphics Metafile ISO / DIS 8632

de courbes. Mais il est difficile de s'étendre vers une reconnaissance d'objets graphiques plus composites, car cela nécessite une analyse poussée de cette liste, et la mise en évidence d'un certain nombre de relations entre les courbes trouvées. Ces difficultés sont essentiellement dues à la pauvreté de la structure créée par les méthodes habituelles. Or notre méthode ne donne pas une structure aussi pauvre ; en effet, l'ensemble des traitements sont effectués sur la structure des composantes connexes, qui est ainsi enrichie au fur et à mesure. Il est donc intéressant de poursuivre le travail de reconnaissance pour tirer profit de toutes les informations glanées et recueillies dans cette structure au cours du processus d'extraction de primitives. Cela est particulièrement vrai quand on dispose d'informations *a priori* sur le graphique à reconnaître, comme nous allons le voir dans la suite.

6.3.5 Objets graphiques de plus haut niveau

Il existe beaucoup d'objets graphiques pouvant être reconnus. Toutefois, certains d'entre eux sont plus fréquemment rencontrés ; ils présentent donc un intérêt particulier, car le fait de les reconnaître peut permettre des gains de place substantiels dans le codage final du graphique.

En premier lieu, on peut chercher des objets graphiques contenus dans une composante connexe. Les exemples les plus immédiats sont les polygones particuliers tels que rectangles, carrés, losanges, triangles rectangles, équilatéraux ou isocèles. La recherche de tels objets dans la structure est immédiate : ce sont des courbes fermées ayant un nombre fixé de segments, ces segments ayant certaines propriétés (parallélisme, angles de valeur déterminée, longueurs identiques, etc.).

On pense aussi naturellement à la reconnaissance de **zones hachurées**. L'idée est simple, et la réalisation n'est guère plus compliquée : il s'agit de trouver sur une composante connexe une courbe fermée, puis un ensemble de courbes ouvertes réduites à un segment, à peu près parallèles et équidistantes, dont les deux extrémités ont une jonction avec la courbe fermée. Si on désire trouver une zone hachurée non incluse dans une courbe fermée, il suffit d'effectuer une recherche de segments parallèles équidistants ayant deux extrémités libres sans points d'intersection avec d'autres courbes. On peut alors créer une courbe virtuelle de couleur blanche joignant toutes ces extrémités, et qui représente le contour de la zone hachurée. L'ensemble peut dans les deux cas être codé par la courbe et le motif d'hachurage, ce qui réduit la place requise et augmente le niveau d'abstraction de la représentation. La recherche de courbes parallèles et équidistantes se fait par un simple parcours de la structure, avec regroupement des courbes trouvées en familles homogènes. On peut donc prévoir un module de reconnaissance de zones hachurées, que l'utilisateur active s'il en ressent le besoin. Ce module est en phase de réalisation à cette heure, et doit être opérationnel pour la fin du mois de juin 1987.

Un autre exemple de primitive graphique est l'**arc de cercle**. Au moment où nous avons présenté l'approximation polygonale, nous avons vu que celle-ci pouvait être étendue à une approximation en

segments et arcs de cercle. Dans ce cas, les étapes d'appariement doivent bien sûr être également étendues pour qu'il soit possible d'apparier deux arcs de cercle, ou un arc de cercle avec un ensemble de segments. La structure contiendrait des branches et des courbes constituées de suites de segments et d'arcs de cercle. Cette approche n'a pas été choisie dans notre système mais n'est pas difficile à concevoir comme une extension des méthodes que nous avons présentées.

Reconnaître des arcs circulaires sur les courbes trouvées par notre méthode est une approche qui se greffe directement sur la structure d'analyse : une série de segments de même taille et ayant une variation de direction continue peut être reconnue comme un arc circulaire. Le rayon, le centre et l'amplitude angulaire de cet arc peuvent être déterminés par des calculs simples. Ce point n'est pas prévu dans le cadre du projet HERODE, mais est tout à fait réalisable sans gros efforts de développement.

A côté de celles que nous venons de citer, d'autres primitives graphiques de plus haut niveau doivent être reconnues sur plusieurs composantes connexes. On peut à ce niveau réexploiter les informations de proximité pour suivre un ensemble de plusieurs composantes. Une application typique est la recherche de la primitive *POLYLINE*[8]. Cette primitive est constituée d'une suite de symboles identiques ordonnés de manière à constituer une ligne de symboles. Par exemple, sur une carte, une suite de croix représente souvent une frontière entre deux pays.

Trouver toutes les primitives de ce genre impliquerait un module de reconnaissance de symboles ; nous allons aborder cet aspect dans la conclusion. Toutefois, il est tout à fait concevable de reconnaître des objets graphiques très simples comme une croix, un petit carré ou même un simple tiret, et de chercher les *polylines* en exploitant les informations de proximité et de distance entre composantes. Il suffit en effet de chercher parmi les composantes proches celle qui est de la même forme et de prolonger la ligne par là. Une fois de plus, une telle reconnaissance permet une représentation plus compacte et plus abstraite : un *polyline* est représenté par le symbole élémentaire, par la courbe formée et par la distance entre deux symboles sur cette courbe.

Rappelons toutefois la difficulté que représente la frontière entre une ligne de symboles et du texte répétitif : la segmentation entre texte et graphique a très bien pu reconnaître un *polyline* bien droit comme un bloc de texte. D'ailleurs, dans son état actuel, elle détecte déjà et étiquette de manière particulière des lignes tiretées horizontales et verticales. Une solution à ce problème serait la réalisation d'un module d'arbitrage qui décide, après une phase de reconnaissance de primitives graphiques, de symboles et de caractères, de remettre en cause la segmentation originale et de reconsidérer comme graphique, par exemple, des objets qui dans un premier temps étaient reconnus comme de type texte. Toutefois, la reconnaissance de caractères ne faisant pas partie des objectifs de HERODE, cet arbitrage n'a pas été implanté.

---

8 voir annexe sur CGM

6.4 Implantation physique et programmation

Le projet HERODE doit aboutir en automne 1987 à un système expérimental où les différents modules seront intégrés. Ce système a été développé pour des stations de travail de type **SUN**, fonctionnant sous le système d'exploitation **UNIX**. Ces stations de travail disposent des outils de base pour construire un système fortement interactif, notamment grâce au multifenêtrage et à l'utilisation de la souris. L'utilitaire ***suntools*** permet de créer et d'utiliser un ensemble d'outils (*tools*) représentés par des icônes. La possibilité d'ouvrir des fenêtres graphiques permet en particulier l'affichage aisé des résultats des différentes étapes de l'extraction de lignes, ce qui est indispensable en phase de mise au point.

L'environnement de programmation disponible sous UNIX sur ce genre de station de travail est aussi très satisfaisant. Le langage de programmation utilisé est le langage **C** ; celui-ci permet d'écrire des programmes très efficaces, et il est le complémentaire naturel du système UNIX, ce qui est un avantage considérable quand il s'agit d'écrire des traitements les plus rapides possibles, en utilisant toutes les possibilités du système d'exploitation. Les outils de développement et de mise au point de programmes disponibles sous UNIX ont aussi facilité le travail.

Toutefois, il restait un point délicat, à savoir l'assemblage des différents modules provenant d'équipes différentes et géographiquement éloignées, ainsi que la création d'un module d'interface utilisateur uniforme pour tous ces modules. On se heurte ici à des problèmes de **communication**, aussi bien entre des modules distincts d'un même système qu'entre l'homme et la machine. Le choix fait dans le projet HERODE pour résoudre ces difficultés est l'utilisation d'un langage à objets, **Objective-C** [13]. Celui-ci est un sur-langage de C implantant le modèle de **SMALLTALK-80**, avec les limitations dues aux contraintes de la compilation, SMALLTALK, lui, étant interprété.

Les différentes parties du module de saisie automatisée de documents étant développées par trois équipes distinctes, l'usage de cette couche orientée objets pour interfacer les parties a été très utile pendant toute la phase de mise au point. En effet, on peut ainsi passer à un niveau supérieur d'abstraction, en considérant les différents traitements comme des objets activés par envoi d'un message[9]. Une fois la configuration à peu près figée, nous avons éliminé cette «couche objets» pour des raisons d'efficacité. Mais bien entendu, Objective-C reste beaucoup employé par les modules fortement interactifs du système complet, c'est-à-dire les éditeurs et l'interface utilisateur.

Notre partie, la reconnaissance de graphiques, a donc été écrite en C. La plupart des fonctions de notre module sont **statiques**, c'est-à-dire locales au fichier où elles sont définies. Seules quelques fonctions aux paramètres bien spécifiés sont accessibles par les autres parties du système ; ce sont elles qui constituent

---

9 il serait peut-être plus correct de parler d'acteurs que d'objets dans ce cas, bien qu'Objective-C ne soit pas un langage d'acteurs

l'interface de notre programme avec le monde extérieur. Les mêmes principes ont été appliqués aux variables globales, qui sont en nombre très réduit. De telles règles de programmation sont fondamentales quand on développe un système de la taille de HERODE, sous peine de se heurter à d'énormes problèmes d'intégration en phase finale.

Les calculs s'effectuent relativement vite : de l'ordre d'une dizaine de secondes pour l'ensemble segmentation texte / graphique et extraction de lignes avec une image de départ de 800 lignes et 800 colonnes, sur un SUN 3/260 (4 *Mips*). On trouvera en annexe des copies d'écran présentant les résultats des différentes étapes de la reconnaissance de primitives graphiques.

Chapitre 7

Conclusion et perspectives

Nous avons vu comment il est possible de trouver l'information graphique dans un document composite en menant toutes les étapes du traitement sur une structure hiérarchique de composantes connexes. Cette structure est créée par la segmentation entre texte et graphique, qui emploie elle-même un grand nombre d'idées originales et donne des résultats remarquables, même en présence de caractères de grande taille ou de texte étroitement mêlé au graphique. L'ensemble des traitements réalisant cette segmentation fera l'objet d'une thèse par Eric Meynieux, notre collègue de TITN.

Tous les algorithmes que nous avons mis au point pour l'extraction de l'information graphique se fondent sur cette structure, qui est ainsi enrichie au fur et à mesure. De ce fait, l'obtention de primitives graphiques n'est qu'une étape, et l'abondance d'informations géométriques, topologiques et structurelles dont nous disposons dans la structure nous permet de continuer sans grosses difficultés à «monter» vers des niveaux de compréhension plus élevés. C'est là un des gros atouts de ce travail : poser des bases saines et robustes dans l'extraction des primitives, en vue de fournir à des systèmes d'interprétation de plans par exemple toutes les informations nécessaires pour obtenir des résultats convenables.

Nous fournissons en sortie de notre module un codage du graphique conforme aux spécifications de la norme ODIF. Les résultats sont satisfaisants pour des images très diverses. Toutefois, un point reste assez sensible : l'extraction des lignes par appariement pose parfois des problèmes, en particulier dans des situations limites où la ligne est presque trop épaisse, ou quand le dessin de départ est très tourmenté (dessin à main levée par exemple). En fait, s'il fut relativement aisé de trouver 90% des lignes, les 9% suivants nous ont forcés à revoir et affiner profondément la méthode pour qu'elle soit solide et rende des résultats satisfaisants dans tous les cas.

Il reste un très faible pourcentage de cas limites où l'extraction de lignes donne des résultats non entièrement satisfaisants ; toutefois, les cas de ce genre que nous avons rencontrés ont pu être résolus en ajustant les paramètres du traitement. On peut donc insister sur la robustesse de la méthode. Une des raisons de cette robustesse est la méthode originale que nous avons choisi d'implanter : s'appuyer non pas sur un squelette de l'image mais sur les contours des composantes connexes. L'appariement de segments répond alors à des règles bien définies, et on évite dans la plus large mesure de subir l'influence des défauts d'un squelette et de l'imprécision liée à la nature discrète d'une image numérique.

A cet égard, notre système se distingue de la plupart des systèmes réalisant la saisie automatique de graphiques. En effet, comme nous l'avons déjà dit, l'approche communément employée fournit une structure de données assez pauvre, telle qu'une liste de branches constituées de segments de droites. Cela est évidemment suffisant pour restituer le graphique avec une précision satisfaisante et pour le coder de manière compacte, mais il est difficile de poursuivre l'analyse vers une représentation plus symbolique comprenant

des objets graphiques plus complexes et des relations entre ces objets. Il est aussi malaisé de reprendre le graphique créé par ces méthodes sous un éditeur graphique ; en effet les objets trouvés ne sont pas forcément ceux qu'on aimerait naturellement manipuler. Ayant gardé le maximum d'informations au cours des étapes, et les ayant structurées de façon cohérente, nous n'éprouvons pas toutes ces difficultés pour poursuivre notre travail vers des niveaux de compréhension plus élevés.

Nous pensons donc que les résultats obtenus, en particulier sur le graphique de type dessin technique, permettent de dépasser le but fixé par HERODE, qui était uniquement de coder du graphique de manière à pouvoir le retraiter par un éditeur graphique, toujours dans un contexte plutôt bureautique. Un champ d'application important des méthodes que nous avons mises au point est celui de la **lecture automatique de plans techniques**. En effet, si tous les bureaux d'étude importants se servent maintenant de la conception assistée par ordinateur (CAO) et en particulier du dessin assisté par ordinateur (DAO), ils disposent d'un très gros volume de dessins d'archive. Il serait alors très utile de pouvoir reconvertir ces dessins dans le format utilisé par le système de CAO, sans avoir à les entrer manuellement à l'aide d'une tablette graphique par exemple. C'est principalement dans cette direction que nous désirons poursuivre notre travail à présent, en visant à élaborer une méthode de **saisie automatique de plans techniques**, couplée à un système de CAO.

Nous entrons ici dans un domaine d'application où la connaissance *a priori* sur le genre de document qu'on traite joue un grand rôle. En effet, pour un type de dessin technique, on peut fixer un certain nombre de symboles et d'objets graphiques composites à reconnaître, suivant le domaine technique sur lequel porte le plan à analyser. Les épaisseurs de traits employées sont aussi souvent normalisées, de même que les polices de caractères utilisées pour le texte. Il est donc possible d'ajuster les paramètres et de faire intervenir le cas échéant des modules spécialisés permettant de reconnaître des objets bien précis.

La structure de données créée par notre méthode est assez imposante, contenant beaucoup d'informations et de relations. Les documents techniques étant souvent de grande taille, le système que nous envisageons doit disposer de moyens de calcul relativement importants, pour pouvoir manipuler cette structure en un temps raisonnable. Toutefois, pour ce qui est de la mémoire, les différentes étapes de traitement peuvent être effectuées sur une composante connexe dès qu'elle est connue en entier, ce qui permet de sauvegarder le résultat de la reconnaissance pour cette composante et d'en débarrasser la mémoire, si celle-ci est trop limitée pour garder toute la structure. Cependant, l'exploitation ultérieure des relations topologiques entre plusieurs composantes est alors rendue plus difficile, puisqu'il faut éventuellement restaurer des informations sauvegardées. Un autre moyen de limiter l'espace mémoire occupé est de remarquer que certaines informations sont redondantes à partir d'une étape et peuvent donc être supprimées ; c'est le cas du codage des contours après l'approximation polygonale et des segments produits par cette approximation après l'extraction des courbes.

Pour réaliser un tel système de lecture de plans techniques, nous avons besoin de mener notre travail suivant trois axes de recherche : la **représentation symbolique** d'un document technique, la **reconnaissance de**

formes, entre autres de symboles, et l'interfaçage avec le système de CAO hôte, c'est-à-dire les problèmes de communication.

Le premier problème est celui de la représentation. Notre travail jusqu'à présent a été effectué sur la structure de données que nous avons présentée en détail dans le chapitre précédent. Cette structure est bâtie sur une arborescence de composantes connexes noires et blanches, ce qui convenait parfaitement pour les buts que nous nous étions fixés. Toutefois, pour représenter un plan technique à un niveau d'abstraction supérieur, il faut changer l'ossature à laquelle l'information est attachée. Deux approches sont envisageables. La première est de représenter un ensemble d'objets graphiques ayant entre eux des relations géométriques et topologiques, mais aussi sémantiques. Ce genre de représentation est plutôt adapté au dessin industriel classique, où la pièce ou l'ensemble est présenté sous différents points de vue et coupes. Le deuxième type de représentation est le graphe : on dispose d'un certain nombre d'objets représentés par des symboles. Ces objets sont reliés par des arcs, qui ont éventuellement des attributs tels que l'épaisseur de trait, l'orientation ou même la signification sémantique de la liaison. Cette représentation est idéale pour des documents tels que les schémas de câblage électrique, les circuits électroniques ou logiques, les cartes de réseaux (électricité, eau, gaz, assainissement, ...), les plans de tuyauteries ou autres conduites, etc.

Dans les deux cas, il est avantageux d'utiliser des outils symboliques pour l'implantation physique de la représentation et sa manipulation. Nous nous orientons donc vers l'emploi de systèmes alliant un langage de manipulation symbolique tel que LISP à un formalisme objet pour la représentation des informations.

Un autre axe de recherche est celui de la reconnaissance de symboles. Beaucoup de méthodes diverses ont été développées pour effectuer ce genre d'opération, particulièrement sur le domaine restreint de la reconnaissance de caractères. Pour reconnaître un symbole, c'est-à-dire décider qu'il appartient à une classe déterminée, on dispose essentiellement de deux types d'informations : les informations structurelles et les informations d'ordre statistique. Certaines méthodes s'appuyent principalement sur un seul de ces types d'informations, ce qui donne naissance à des méthodes structurelles et syntaxiques d'un côté [9], et à des méthodes purement statistiques de l'autre [11]. Toutefois, il est tout à fait possible de se servir de méthodes mixtes, tirant profit de toutes les informations disponibles. Cela peut se faire par exemple en effectuant une première classification du symbole à reconnaître suivant des critères structurels ; ensuite, son affectation définitive se fait par classification statistique.

Le problème de reconnaissance de symboles n'est qu'un cas particulier de la reconnaissance de formes bidimensionnelles ; on peut donc *a priori* envisager l'emploi de techniques générales telles que la génération et l'évaluation récursives d'hypothèses, comme le fait un système tel que HYPER [5]. Toutefois, de tels systèmes risquent d'être lourds à mettre en oeuvre, et il peut être plus judicieux de tenir compte des particularités des symboles.

Dans les problèmes de reconnaissance de symboles techniques, une solution souvent adoptée est de représenter le symbole par un graphe, ses points particuliers étant les noeuds du graphe et les traits joignant

ces points les arcs. La reconnaissance du symbole peut alors se faire par sa mise en correspondance avec un modèle. C'est ce que font entre autres Groen *et al.* [20], dans un système de reconnaissance de schémas électriques et de circuits logiques. Les symboles sont délimités ; ils sont reliés par des lignes droites ou brisées. Chaque symbole est représenté par un graphe comportant une liste de noeuds et une liste d'arcs reliant ces noeuds. A chaque noeud sont associés des attributs tels que la position relative du noeud, son type (extrémité, point anguleux, point de jonction), la liste des segments attachés, etc. Les arcs reliant les noeuds sont les segments reliant les points caractéristiques du symbole ; ils ont des attributs comme leur longueur, leur courbure, les noeuds qu'ils relient.

Dans une première phase, on construit des modèles de classes, où sont définis pour chaque noeud des paramètres statistiques tels que le degré de confiance du noeud (apparaît-il dans toutes les instances de la classe ?), sa position moyenne, l'histogramme du nombre d'arcs y aboutissant, etc. La reconnaissance se fait ensuite par mise en correspondance probabiliste : pour chaque modèle existant, les quatre noeuds les plus stables et les plus fréquents sont choisis, et on essaye de leur faire correspondre au mieux des noeuds du symbole à reconnaître. Le modèle donnant le meilleur score est choisi comme classe d'appartenance du symbole. Par ailleurs, les symboles sont reliés entre eux par des lignes ; ces lignes viennent compléter la structure de données et fournissent donc une description logique du schéma traité.

De plus, si on se donne un certain nombre de règles sur les liaisons valides et invalides entre deux symboles, on peut remettre éventuellement en cause la reconnaissance d'un symbole si elle aboutit à une incohérence. C'est ce que proposent Murase et Wakahara [37], dans un système de saisie automatisée d'organigrammes informatiques tracés à la main. Leur reconnaissance de symboles repose aussi sur la notion de graphe, et par ailleurs ils analysent la structure logique trouvée d'un point de vue sémantique, en appliquant certaines règles de cohérence propres au domaine d'application. Cela permet la correction de certaines erreurs, par retour arrière sur la reconnaissance de symboles quand la figure reconnue est incohérente.

Nous voyons donc qu'un certain nombre de systèmes expérimentaux ont exploré avec succès ces différentes voies en reconnaissance de symboles. Ils partaient généralement d'une structure beaucoup plus pauvre que la nôtre. Nous sommes donc convaincus qu'il y a beaucoup à faire dans ce domaine et que des résultats très intéressants peuvent être obtenus.

Le troisième problème qui se pose est celui de la communication. Tout d'abord, il peut s'agir d'interfacer le système sur un produit de CAO existant. Dans ce cas, c'est principalement un problème d'adaptation à une norme spécifiant la manière de représenter les objets graphiques trouvés. Mais un autre aspect, bien plus intéressant à notre avis, est la communication avec l'homme. En effet, une application d'un système tel que nous l'envisageons est la saisie directe de dessins à main levée, avec restitution «propre» des objets graphiques représentés. Cela permettrait à un concepteur par exemple d'esquisser ce qu'il désire réaliser, et de convertir directement cette esquisse en plan technique sur lequel il peut travailler. Jansen et Krause [25], ainsi que Murase et Wakahara [37] que nous avions déjà cités, proposent des systèmes expérimentaux

fournissant de tels outils. Une fois de plus, la connaissance *a priori* joue un rôle capital, en permettant de fixer les genres de symboles qui peuvent exister.

En abordant le domaine sous un autre angle, on aboutit au **contrôle** de schémas techniques, qui est en quelque sorte complémentaire des applications précédemment envisagées. Sur un schéma de câblage électrique par exemple, on cherche dans ce cas à détecter des liaisons incohérentes. Pour cela, on dispose d'un certain nombre de règles sur la validité des liaisons entre deux symboles différents. Un tel système doit bien sûr être capable de reconnaître les symboles pouvant apparaître, ainsi que les courbes les reliant, ce qui est donné immédiatement par notre structure d'analyse. Un domaine d'application est la vérification de circuits logiques, où le grand nombre d'interconnexions rend le contrôle visuel très ardu.

Nous avons fixé des buts qui dépassent la reconnaissance de symboles : il s'agit en fait d'interpréter à un niveau sémantique le document technique, en partant de son image acquise par un scanner par exemple. Nous comptons utiliser dans une large mesure les méthodes d'extraction de primitives graphiques mises au point dans le cadre du projet HERODE. Par ailleurs, nous pourrons aussi tirer profit de la compétence existant au CRIN dans le domaine de l'interprétation de dessins, grâce au système **MIRABELLE** [31] [35]. Bien qu'ayant des objectifs très différents, ce système a permis d'explorer l'utilisation de méthodes syntaxiques pour reconnaître les objets réels représentés par des dessins à main levée. On y insiste aussi sur la nécessité de garder le maximum d'informations en cours d'analyse. Une autre nécessité fondamentale est la **flexibilité** de la méthode : elle doit être capable de prendre en compte de nouvelles informations *a priori* sur le domaine traité, d'adapter sa stratégie d'analyse à la connaissance dont on dispose, que cette connaissance vienne de l'extérieur ou des étapes antérieures du système. MIRABELLE restait très pauvre en primitives élémentaires traitées, puisqu'on ne disposait que de segments de droite. Une fois de plus, notre structure de données et les primitives graphiques trouvées nous permettent d'espérer d'encore meilleurs résultats, sur des images de départ bien plus complexes que celles traitées par MIRABELLE.

Nous voyons donc s'ouvrir de vastes champs d'application, surtout dans le domaine des documents techniques. La lecture automatique de documents quelconques, et en particulier de plans techniques, nous semble être prête à devenir une réalité. C'est une application qui intéresse beaucoup de sociétés, et qui devrait être disponible comme produit industriel dans quelques années. Les produits disponibles actuellement sur le marché se limitent à squelettiser, vectoriser et éventuellement extraire les cartouches et le texte sur des critères de taille de composantes connexes. Nous sommes capables de fournir plus d'informations, comme les zones hachurées, les polygones de forme bien définie, les lignes tiretées ou plus généralement les *polylines*. Mais surtout, nous disposons d'une structure suffisamment riche pour permettre encore plus d'interprétation, comme des traits d'axe et de cote ou des symboles complexes. Enfin, nous avons défini nos perspectives pour la suite de ce travail, et nous avons vu qu'il est tout à fait possible de réaliser un système permettant la saisie automatique et l'interprétation à un niveau sémantique élevé de graphiques, et en particulier de plans techniques.

Annexe A

Résultats

Les pages suivantes sont des copies d'écrans SUN illustrant le déroulement de notre système. La figure 24 est l'image de départ d'un exemple «d'école» ; les primitives graphiques trouvées sont indiquées en figure 25. On remarque en particulier que la tache noire est représentée par son contour ; l'information d'épaisseur affichée dans ce cas n'a aucune signification sémantique.

La figure 26 est particulièrement intéressante pour les résultats de la segmentation. En figure 27 on voit que le texte a été correctement extrait, y compris les formules et caractères isolés. De plus les caractères ont été si possible regroupés en mots, lignes et paragraphes, ou associés quand ils appartiennent à une même entité, comme c'est le cas pour les formules dans la figure ; en fait, la structuration physique du document est bien extraite par la segmentation. La figure 28 montre les objets graphiques trouvés ; on peut remarquer que les lignes pointillées sont trouvées comme des objets particuliers, qui ne seront pas soumis au traitement général d'extraction de lignes. La courbe de fonction ne fait pas partie de ce graphique : elle est considérée par le système comme trop compliquée pour être codée en termes de primitives graphiques, et sera donc représentée par un code de genre fac-similé, au même titre que l'écriture manuscrite par exemple. Il n'y a donc que 3 primitives graphiques à reconnaître pour le système d'extraction de courbes : 2 lignes droites et un rectangle. C'est ce qu'illustre la figure 29.

La troisième image, représentée en figure 30, illustre les heuristiques employées pour trouver les primitives les plus «naturelles» : nous voyons en figure 31 que le système trouve le rectangle englobant et un ensemble de segments de droite, et non une suite de petits rectangles par exemple.

Le quatrième exemple est plus conséquent, et reprend les différentes étapes de notre méthode sur une image comprenant beaucoup de graphique. La figure 32 montre l'image de départ, qui est segmentée en parties texte (figure 33) et graphique (figure 34). Sur ces dernières, on effectue d'abord une approximation polygonale des contours, affichée en figure 35. Les résultats de l'appariement et de l'extraction des lignes sont présentés en figure 36, suivis du calcul des points de jonction en figure 37. Enfin, les figures 38 et 39 montrent l'extraction des primitives graphiques, en cours et terminée.

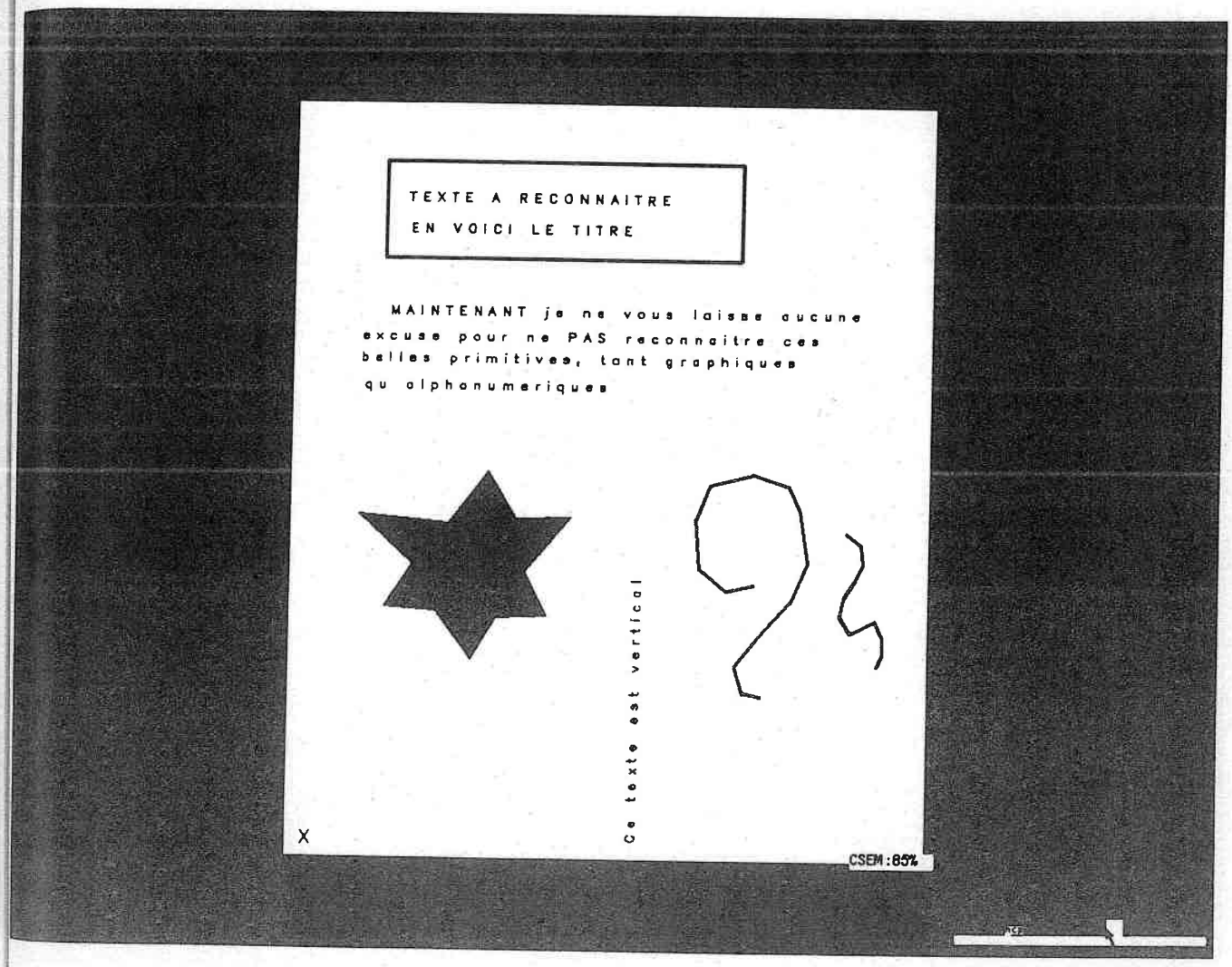


Figure 24. Exemple 1 : image originale

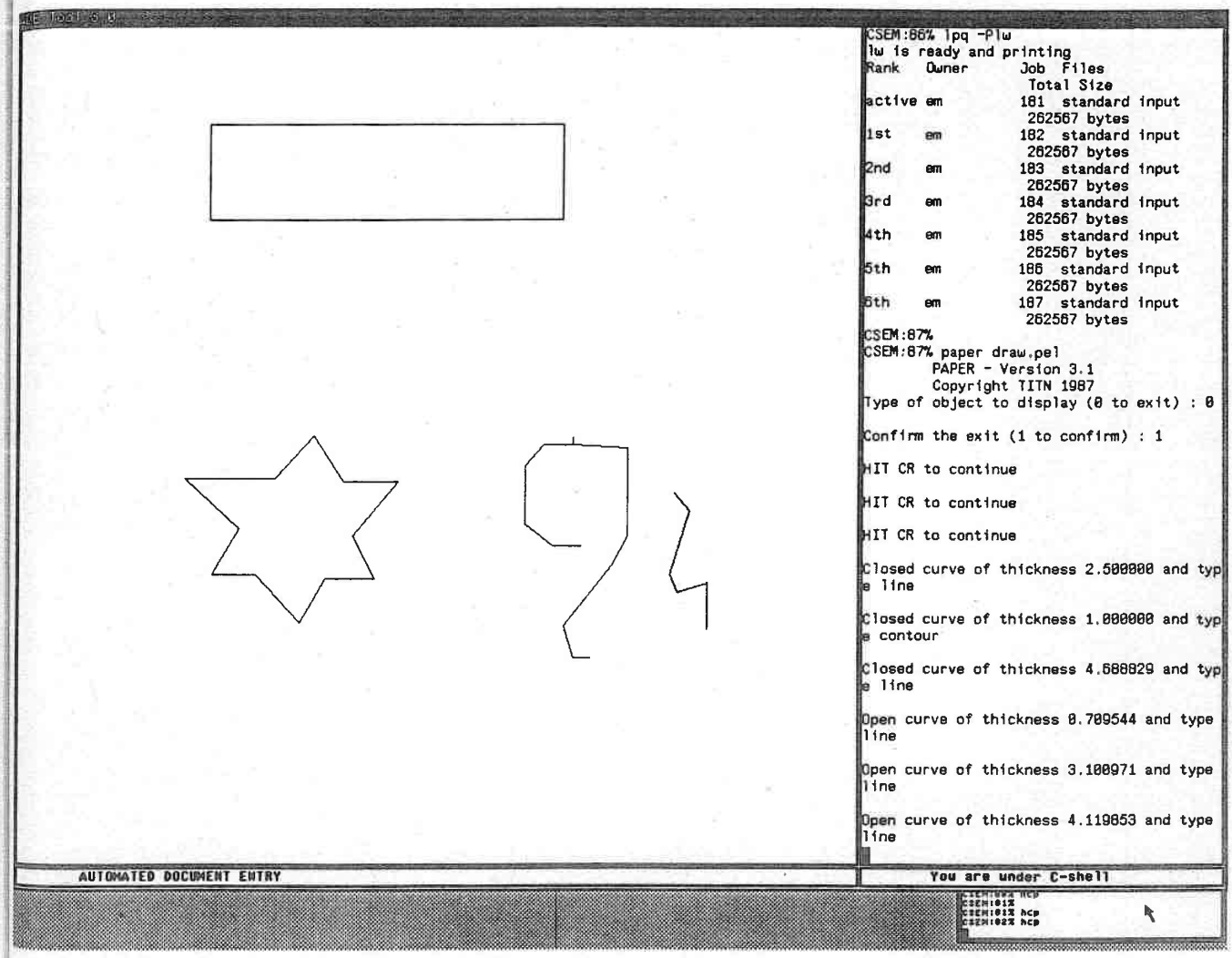


Figure 25. Exemple 1 : primitives graphiques

Cela est d'autant plus valable que $T\Delta f$ est plus grand. A cet égard la figure 2 représente la vraie courbe donnant $|\phi(f)|$ en fonction de f pour les valeurs numériques indiquées page précédente.

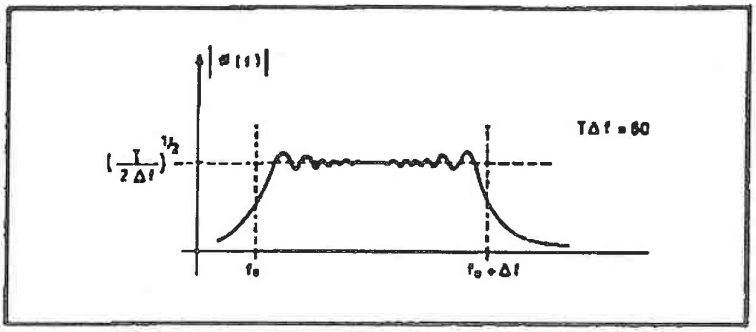


FIG. 2

CSEM : 49%

Figure 26. Exemple 2 : image originale

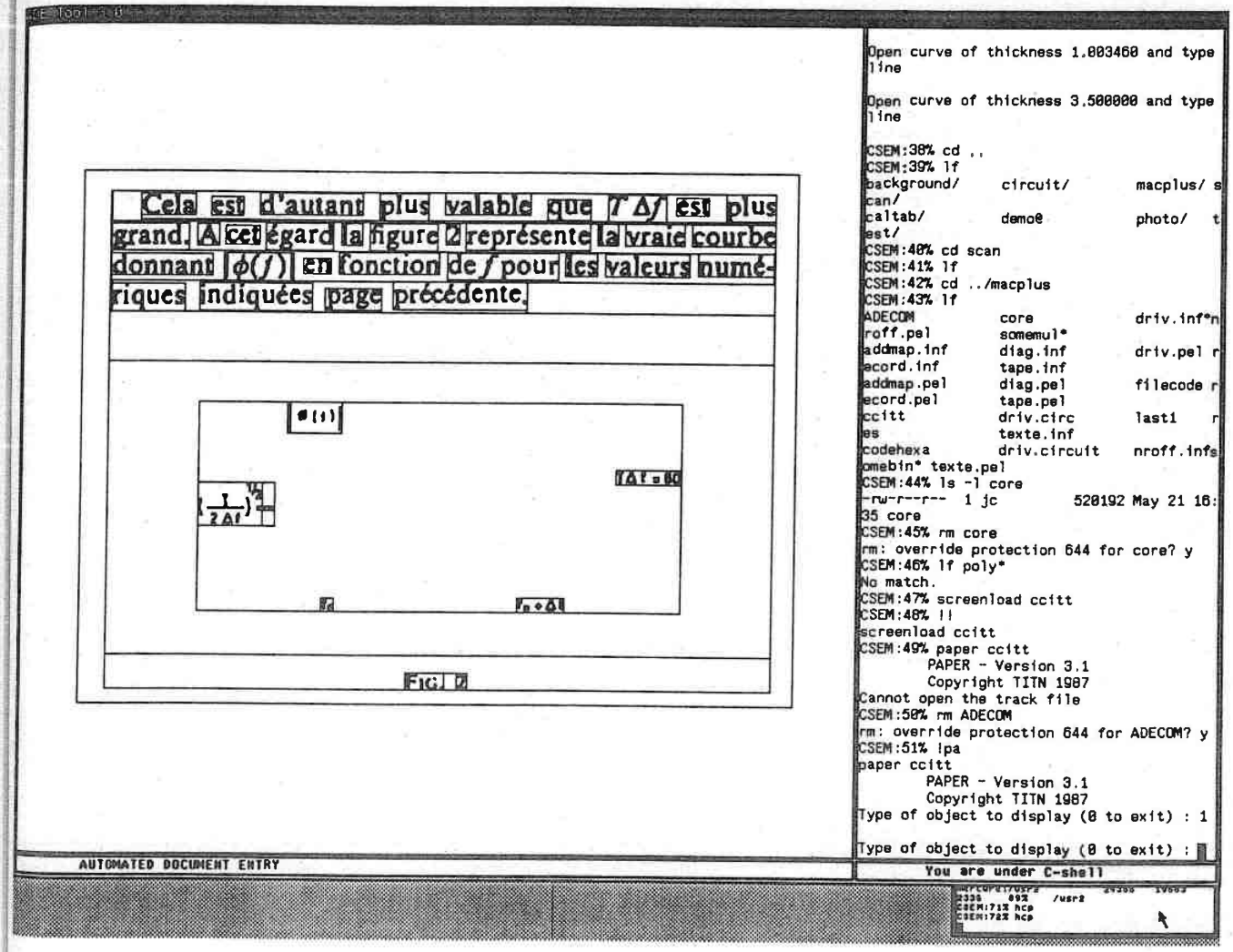


Figure 27. Exemple 2 : parties texte

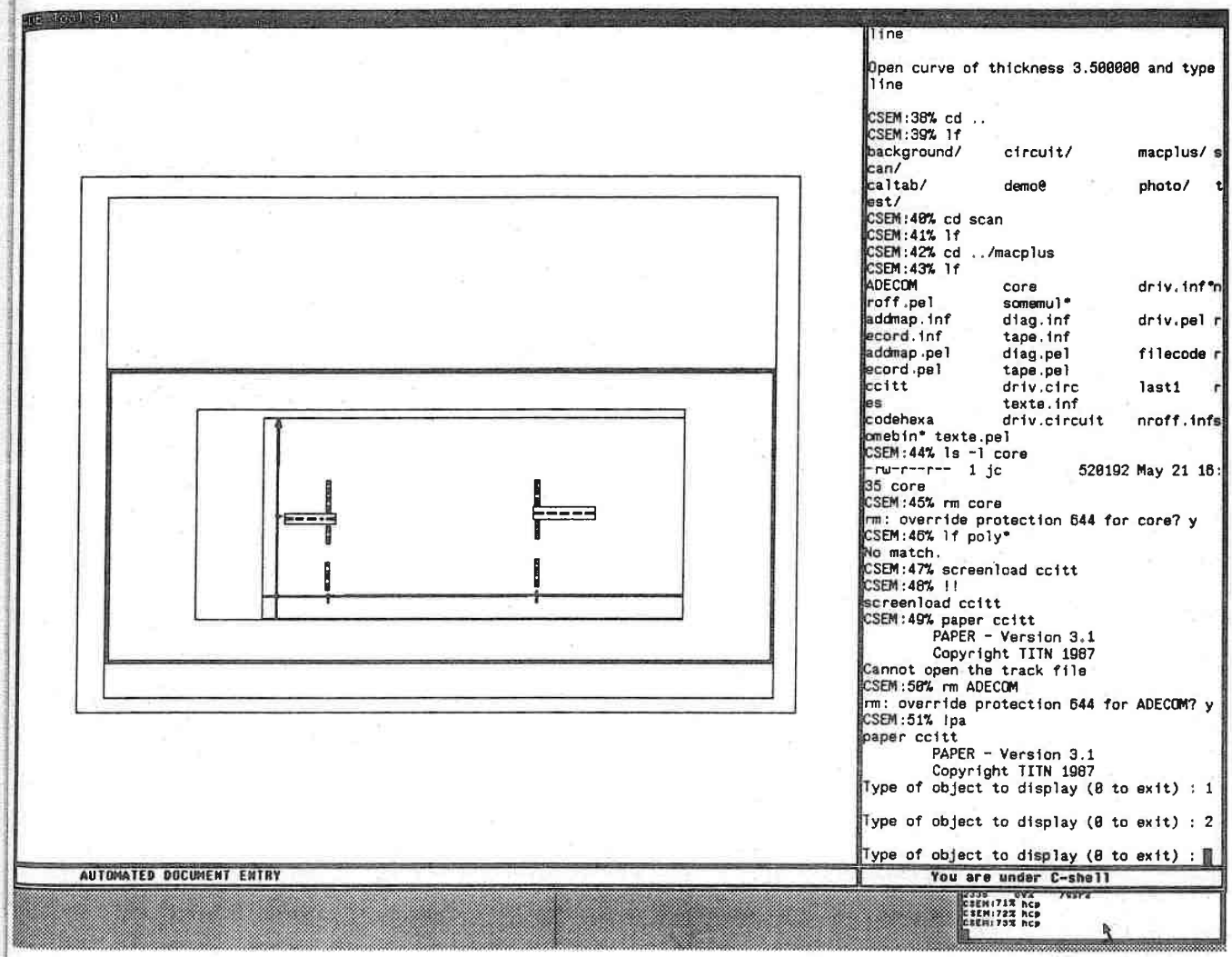


Figure 28. Exemple 2 : parties graphiques

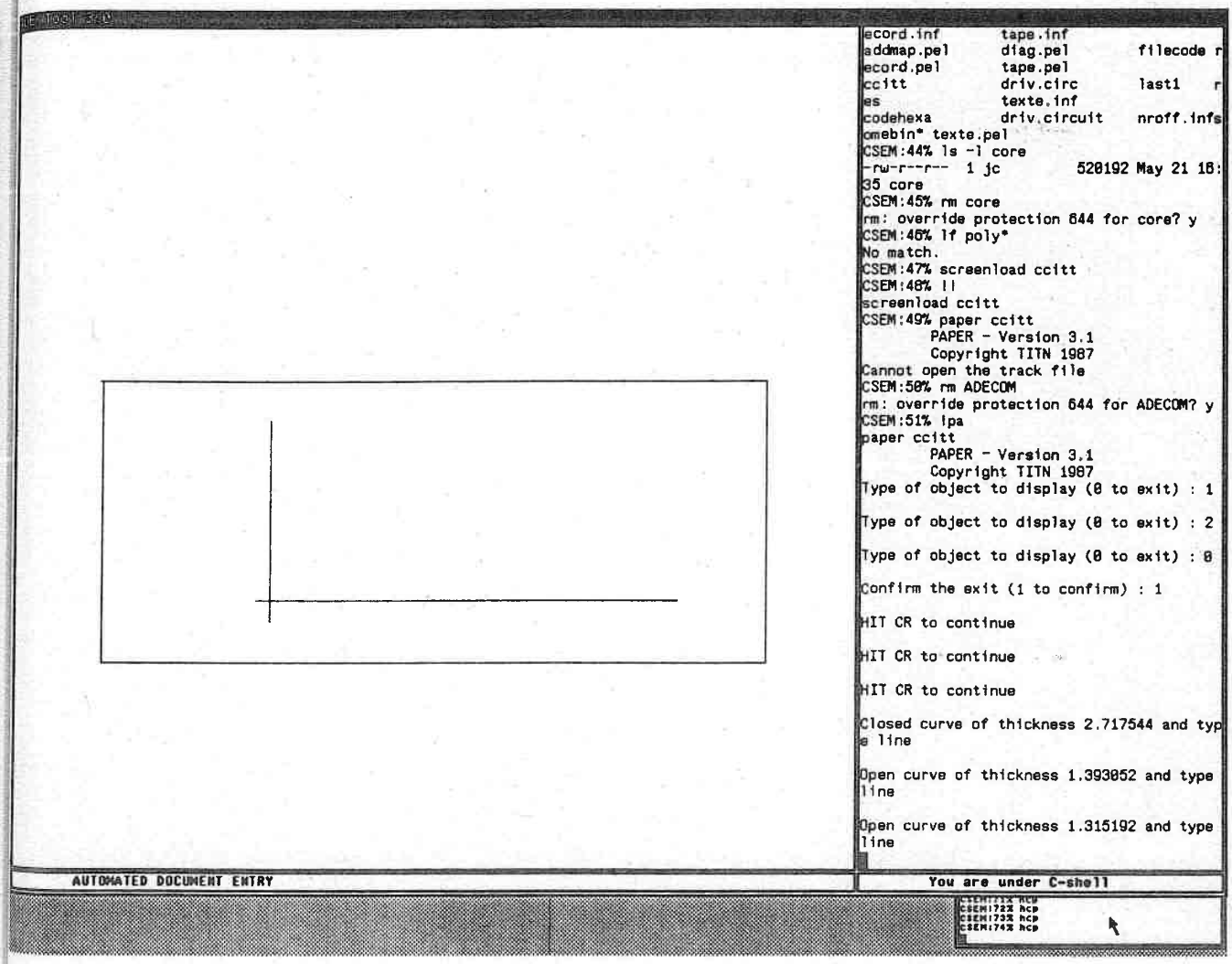


Figure 29. Exemple 2 : primitives graphiques

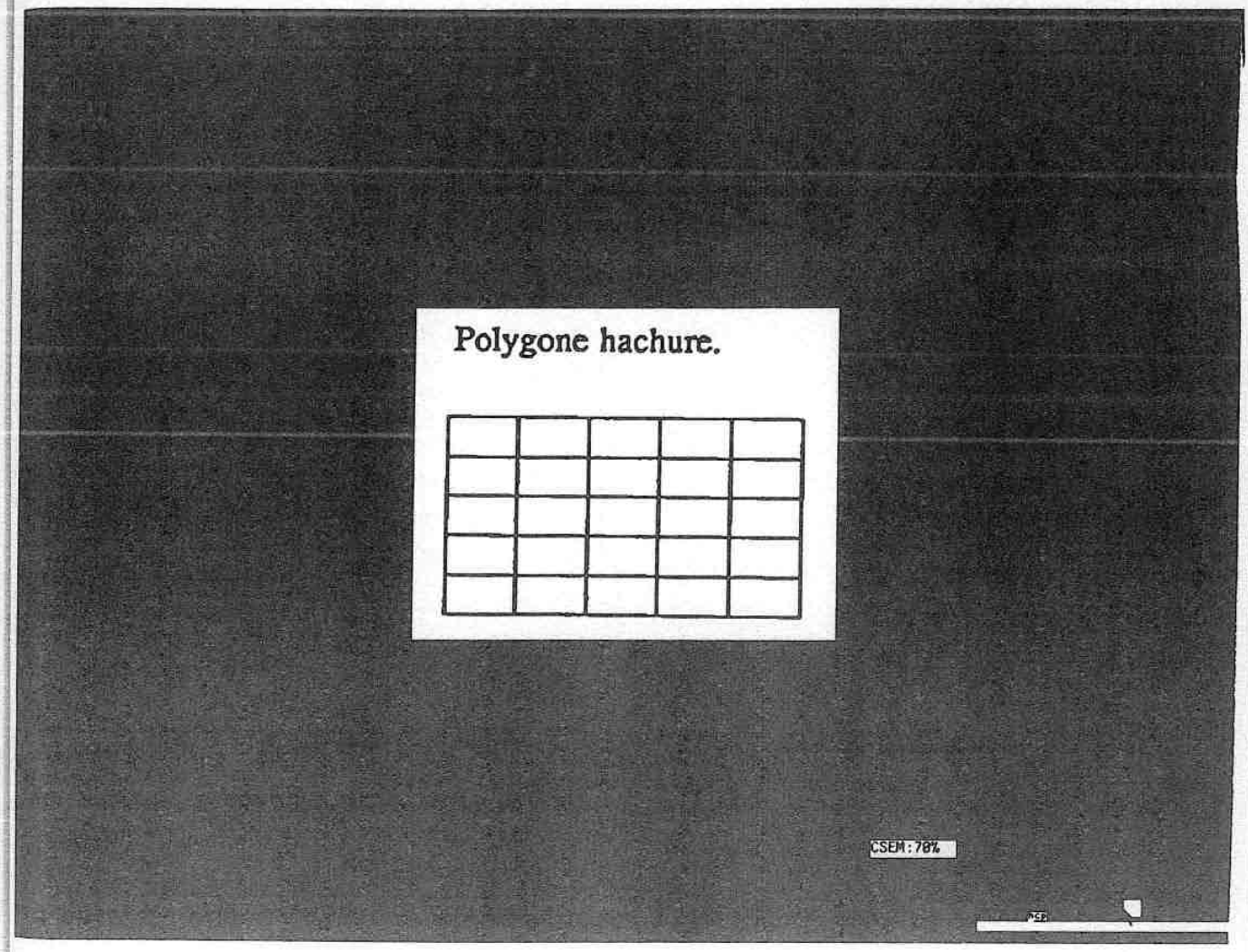


Figure 30. Exemple 3 : image originale

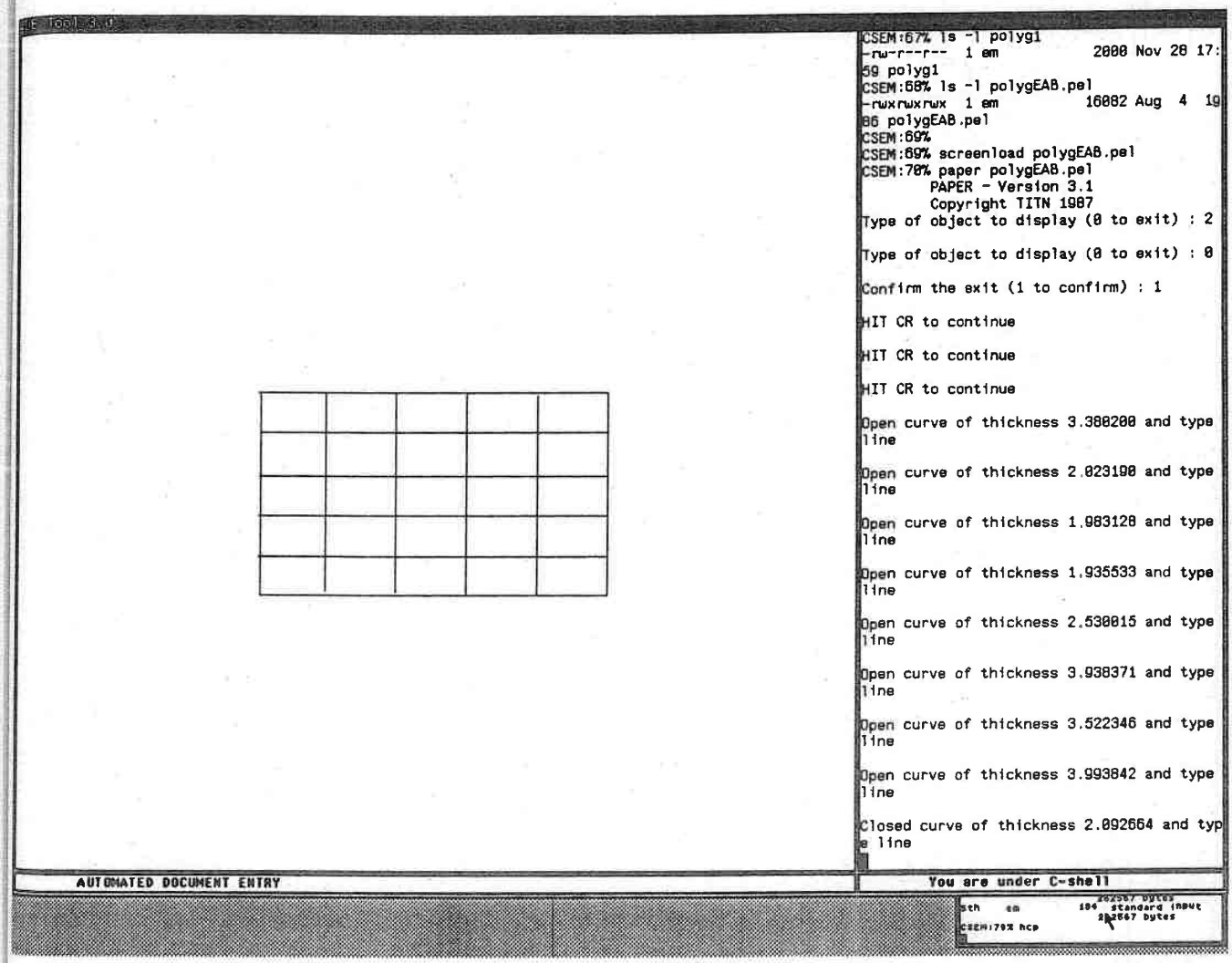


Figure 31. Exemple 3 : primitives graphiques

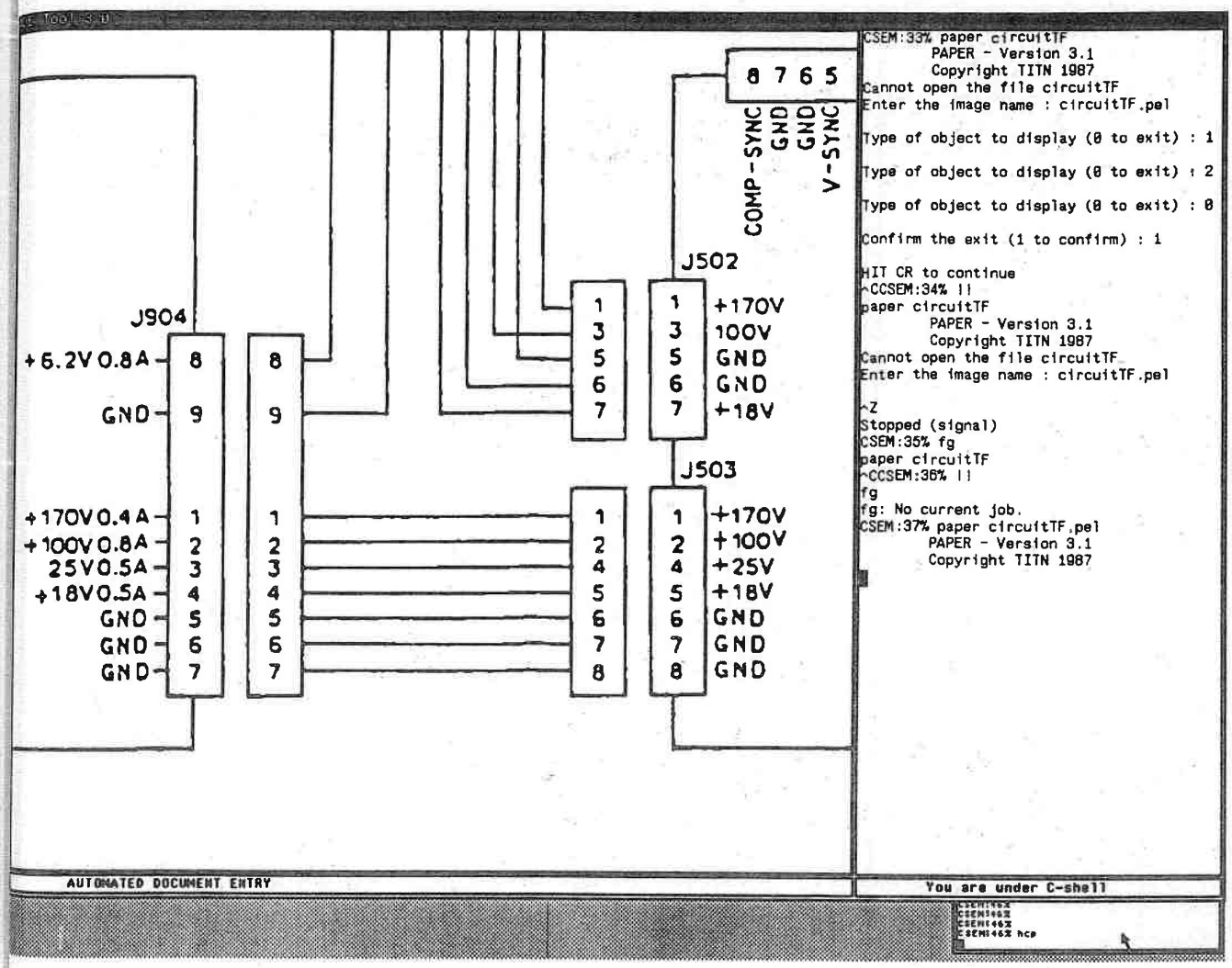


Figure 32. Exemple 4 : image originale

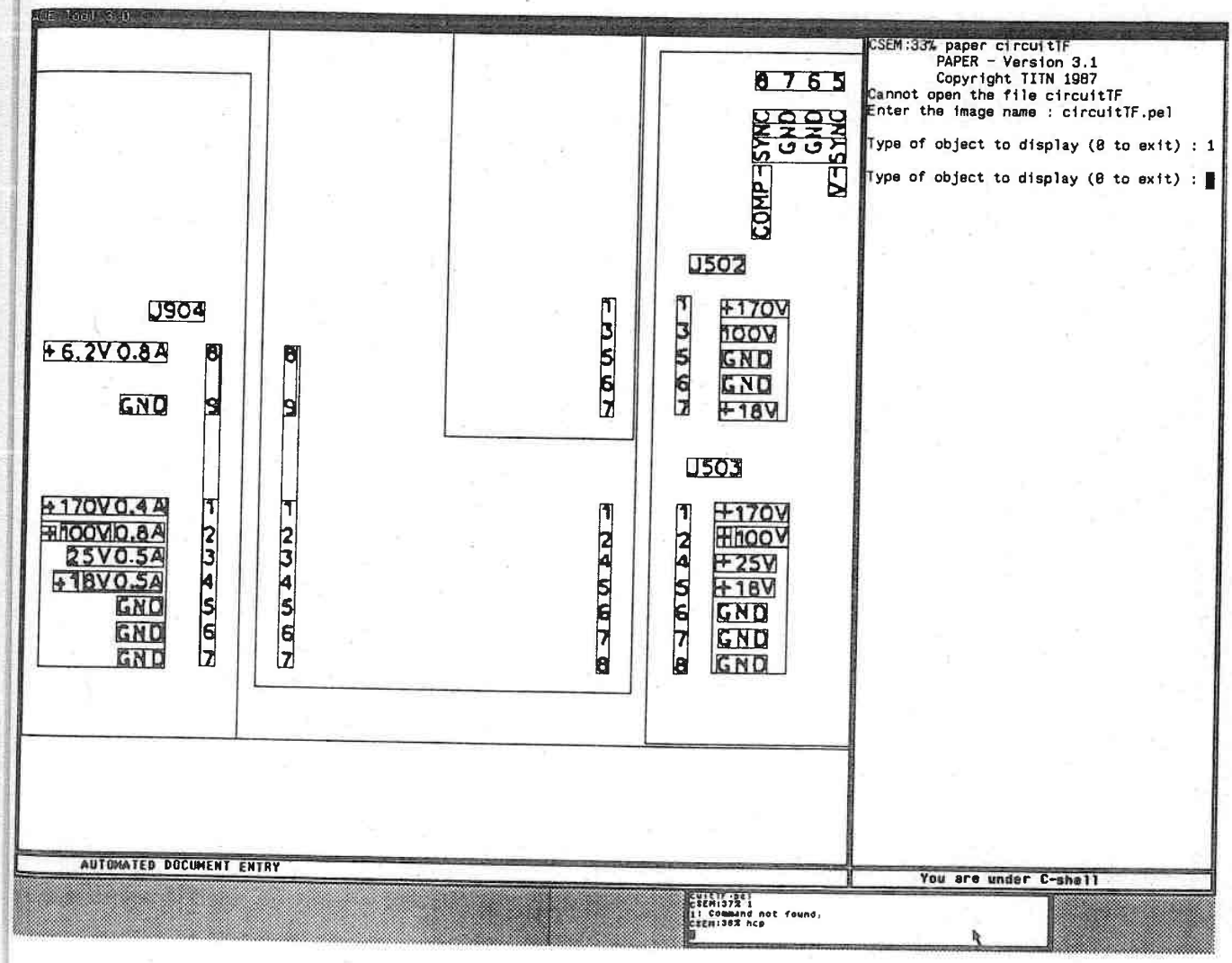


Figure 33. Exemple 4 : parties texte

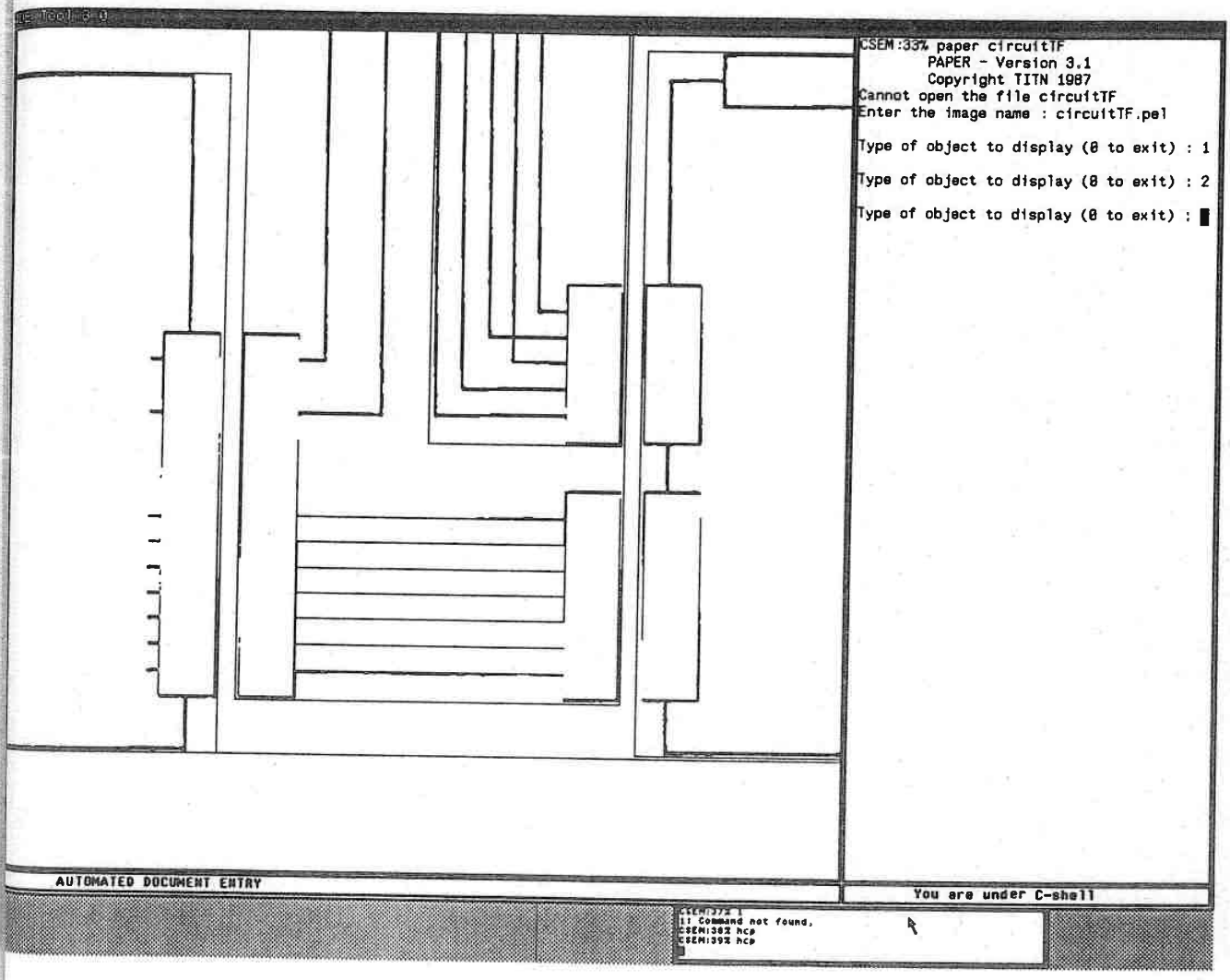


Figure 34. Exemple 4 : parties graphiques

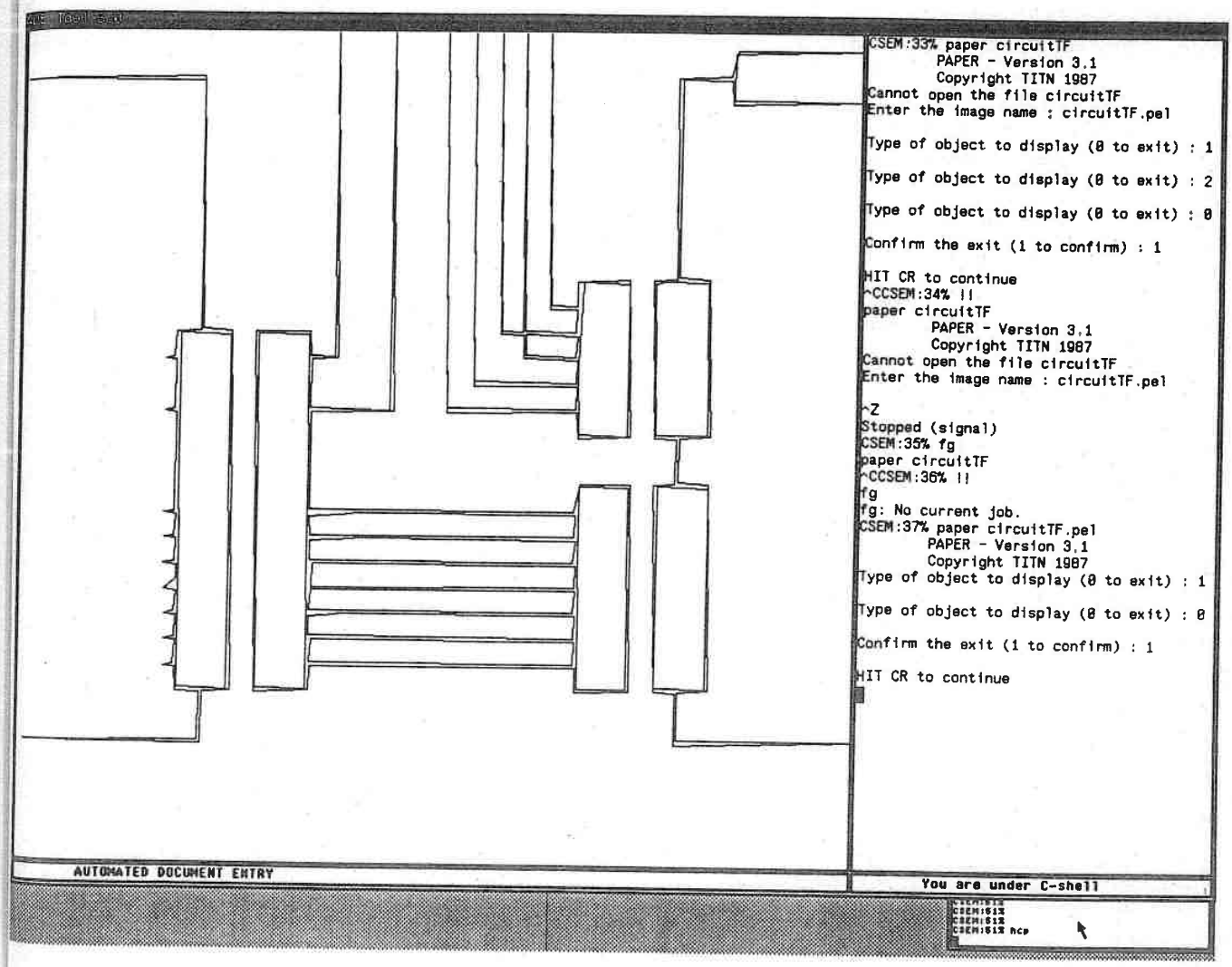


Figure 35. Exemple 4 : approximation polygonale des contours

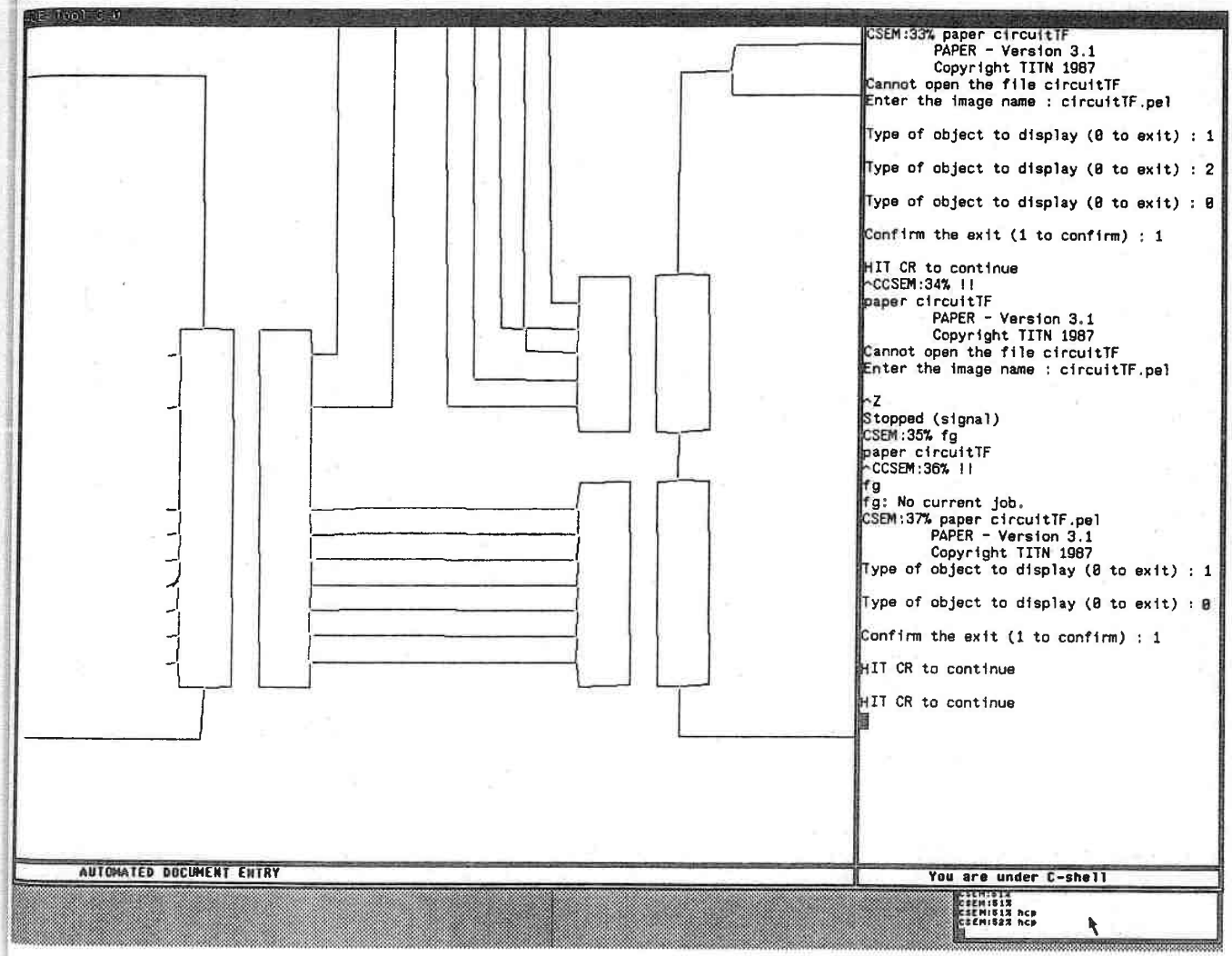


Figure 36. Exemple 4 : extraction des lignes

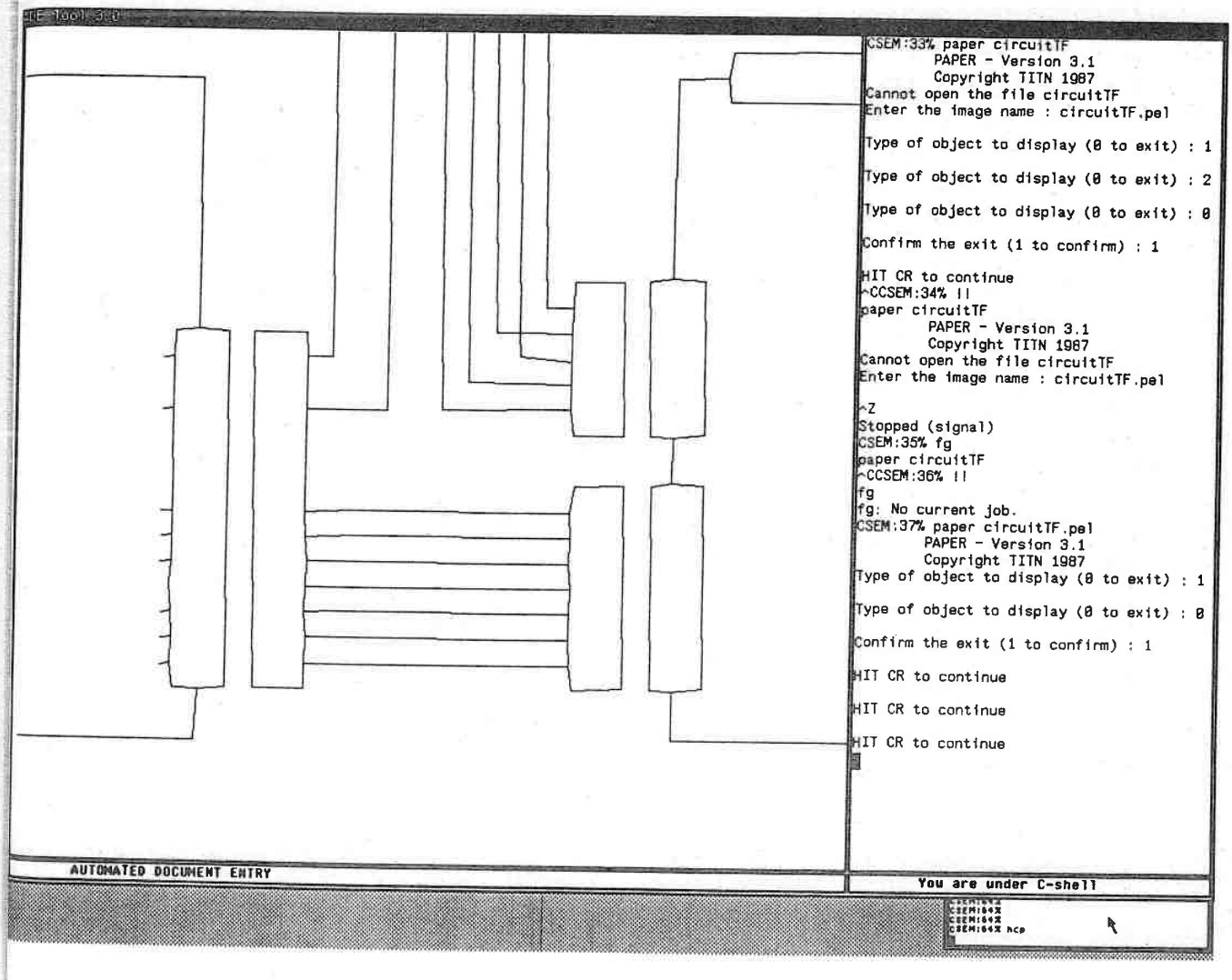


Figure 37. Exemple 4 : traitement des points de jonction

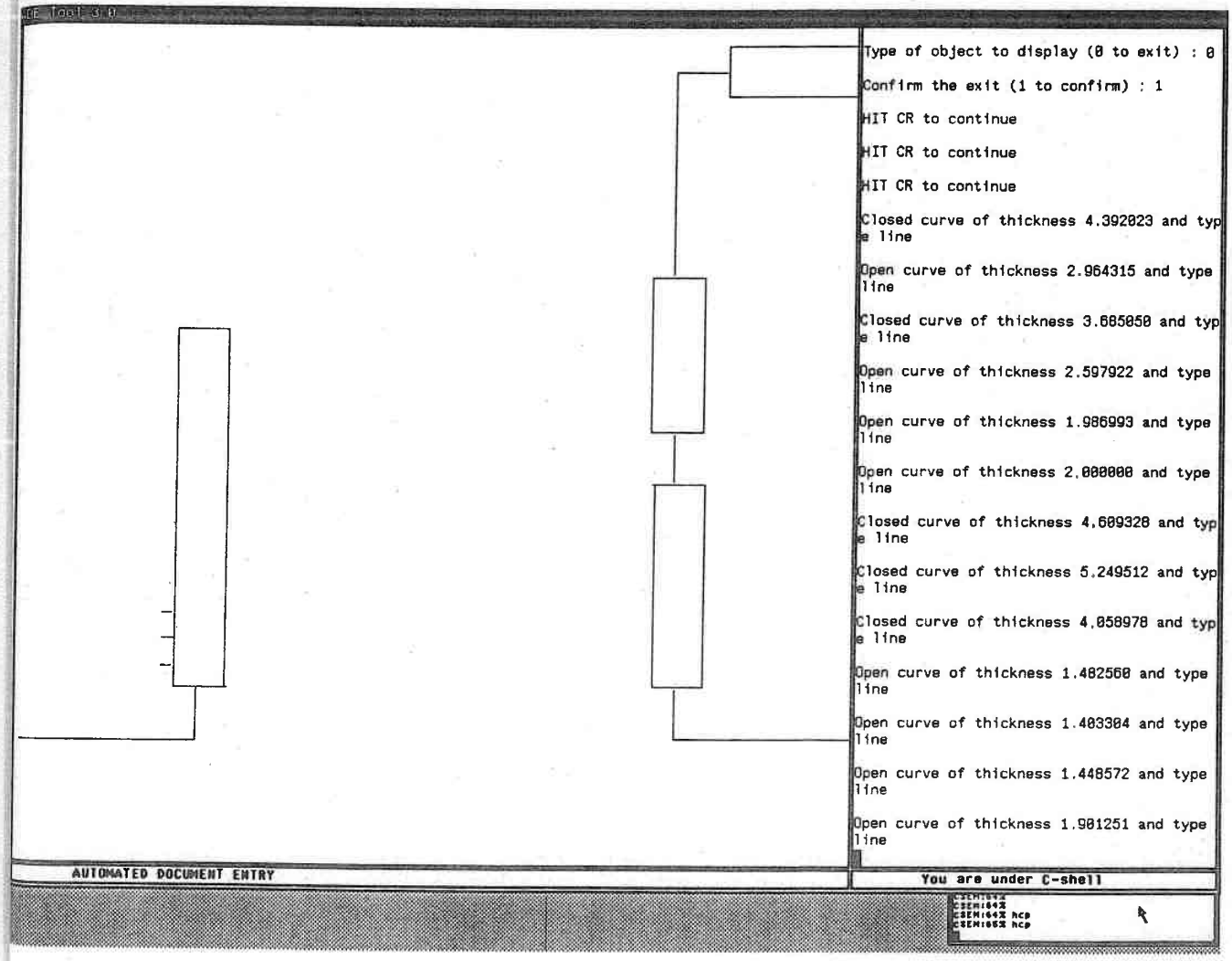


Figure 38. Exemple 4 : extraction des primitives en cours

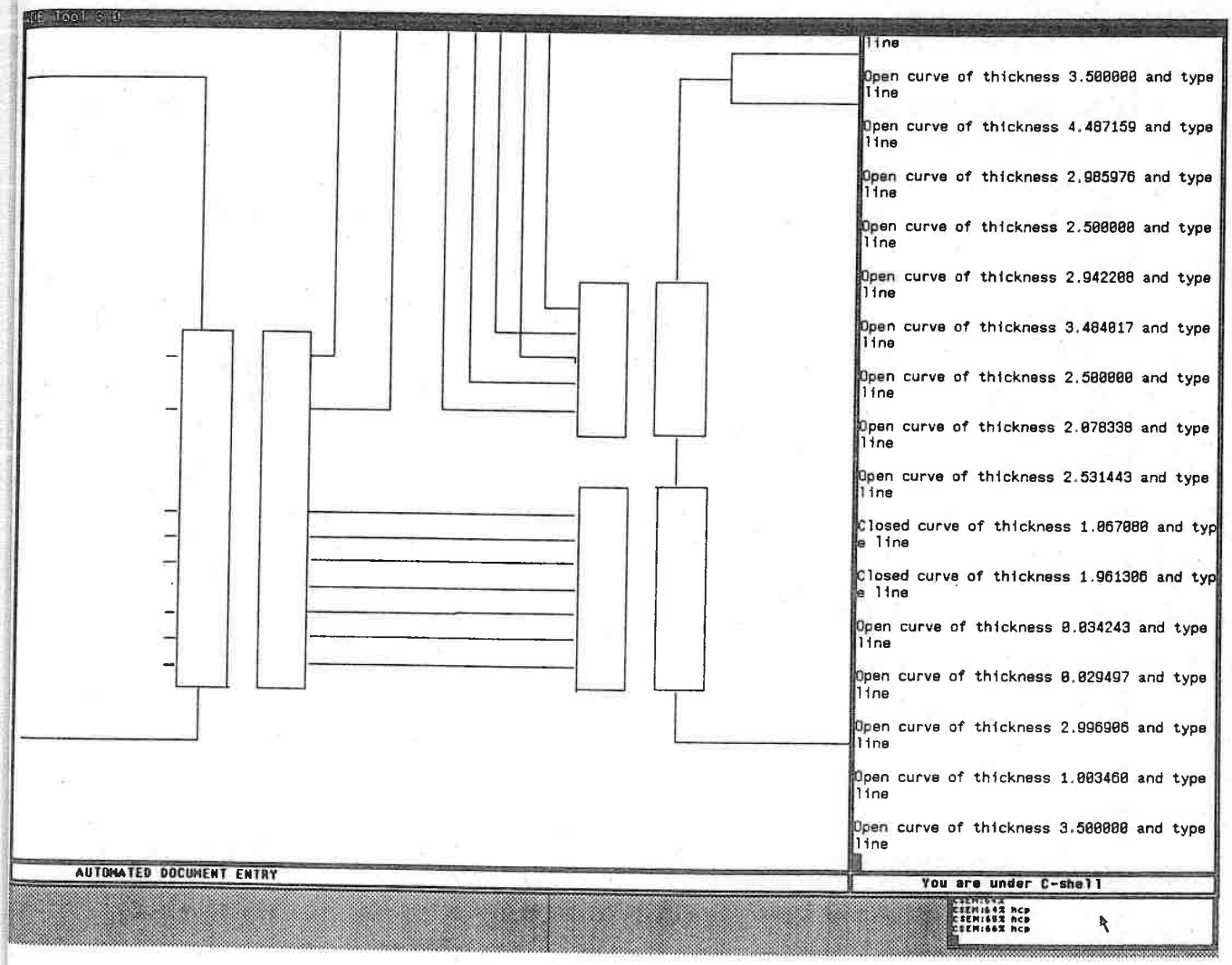


Figure 39. Exemple 4 : primitives graphiques

Annexe B

CGM

La création et la manipulation de graphiques sont des applications très courantes actuellement. On trouve des systèmes graphiques aussi bien sur des micro-ordinateurs que sur des stations de travail ou des machines plus puissantes. Devant la multiplicité de ces systèmes se posent de gros problèmes de normalisation. Nous donnons dans cette annexe un rapide aperçu des différentes normes élaborées ou en cours, en insistant plus sur la norme CGM, choisie par le standard ECMA 101 pour le codage des parties graphiques d'un document. Une présentation plus détaillée tout en restant générale peut être trouvée dans l'article de Bono [10] par exemple.

Les travaux de normalisation portent sur deux aspects des systèmes graphiques. Le premier est l'interface avec les **programmes d'application** : bien que les différents systèmes puissent disposer de représentations internes et de particularités très diverses, il est intéressant de définir un ensemble de fonctions de plus haut niveau qui soient normalisées. Ainsi, un programme d'application pourra être écrit dans une large mesure en utilisant ces fonctions, ce qui améliore sa portabilité d'un système à l'autre. Il existe principalement trois projets de normes pour l'interface avec les programmes d'application :

PHIGS — La norme PHIGS (*Programmer's Hierarchical Interactive Graphics System*) a été conçue pour normaliser les programmes d'application de systèmes importants, comme en CAO/FAO par exemple. C'est une norme pour des systèmes hautement interactifs et nécessitant des moyens de calcul relativement importants. Une implantation de PHIGS sur micro-ordinateur n'est pas concevable à l'heure actuelle, ces machines n'ayant pas suffisamment de puissance de calcul.

GKS — GKS[10] (*Graphical Kernel System*) est constitué d'à peu près 200 routines standards permettant au programmeur de créer et de manipuler des objets graphiques. Des implantations plus ou moins complètes de GKS ont été réalisées sur de nombreux systèmes, et dans plusieurs langages (Fortran, Pascal, Ada, C). En particulier, l'éditeur graphique développé pour le projet HERODE est un éditeur GKS écrit en C. On trouve aussi des systèmes GKS sur micro-ordinateurs, ce qui en fait un standard intéressant pour de nombreuses applications.

GKS-3D — GKS-3D[11] est l'extension de GKS au graphique tridimensionnel, c'est-à-dire la représentation «fil de fer» de volumes. Le besoin de cette norme se fait sentir avec l'apparition de stations de travail graphiques disposant dès le départ de capacités de visualisation et de manipulation d'objets tridimensionnels.

---

[10] GKS : ISO 7942

[11] GKS-3D : ISO DP 8805

Le deuxième aspect sur lequel portent les efforts de normalisation est celui de la sortie des objets graphiques créés par le système. Ces objets peuvent par exemple être affichés sur un écran graphique ou sur un terminal spécialisé, imprimés sur une imprimante laser, transmis par un système de vidéotexte, ou stockés dans un fichier. Afin de limiter la somme de travail nécessaire pour connecter un nouveau périphérique au système graphique, il est intéressant de pouvoir représenter les objets graphiques sous un format d'échange commun ; il suffira ensuite d'écrire les *drivers* qui traduisent ce format standardisé dans le «langage» spécialisé du périphérique. C'est ce qu'on appelle un périphérique virtuel (*virtual device*). De plus, quand on stocke le graphique sur disque, il est utile qu'il soit codé de manière normalisée ; ainsi, il pourra être relu par un autre système graphique, à condition que ce système dispose d'un traducteur entre le format de stockage standard et sa représentation interne.

Deux normes concernent cet aspect des systèmes graphiques :

CGM — Le concept de métafichier (*metafile*) est celui du stockage sur disque indépendant de la représentation interne. GKS définit déjà un *metafile*, essentiellement composé de la trace des différentes opérations GKS effectuées pour créer le graphique qu'on veut stocker. Toutefois, dans un système très interactif, un tel codage n'est pas forcément compact, si de nombreuses opérations de modification ont été faites. On désire donc pouvoir décrire de manière compacte l'état du graphique à un moment donné. C'est ce que fait CGM[12] (*Computer Graphics Metafile*), en se servant d'un ensemble de primitives standards, munies éventuellement d'attributs. Nous donnons par la suite la liste des éléments de CGM.

CG-VDI — CG-VDI (*Computer Graphics Virtual Device Interface*) est une spécification fonctionnelle de l'échange de données normalisées entre un système ayant des fonctions graphiques et un périphérique. C'est une extension de CGM, et on y retrouve à peu près tous les éléments de celui-ci. Mais il contient plus que CGM ; il permet entre autres de créer des primitives graphiques, leur attacher des attributs, spécifier de manière normalisée les caractéristiques d'un périphérique donné, gérer des groupes de primitives appelés ***segments***, et lire du graphique en entrée.

Les éléments de la norme CGM

Nous présentons ci-dessous les différents éléments de CGM ; pour cela nous nous sommes fondés sur l'article de Bono précédemment cité. Un *metafile* CGM est formé d'une ou plusieurs images graphiques, appelées *pictures*. Il faut bien sûr des délimiteurs entre ces images, ainsi que d'autres instructions de contrôle. Le *metafile* peut être représenté sous 3 formes :

---

12 CGM : ISO / DIS 8632

- en codage par caractères alphanumériques, ce qui est surtout utile pour la transmission du document d'une machine à une autre,
- en codage binaire, ce qui est la manière la plus compacte de stocker le document localement,
- par du texte en clair (le nom de toutes les primitives *in extenso*), ce qui n'est pas compact, mais en revanche très lisible et facilement éditable.

Nous donnons ici le nom des éléments tels qu'ils apparaissent dans la norme ; ils sont assez explicites pour ne pas nécessiter de longues explications :

1. Les délimiteurs :

   BEGIN METAFILE, END METAFILE
   BEGIN PICTURE, END PICTURE
   BEGIN PICTURE BODY

2. Les descripteurs de *metafiles* (*Metafile Descriptor*) :

   METAFILE VERSION, METAFILE DESCRIPTION
   VDC TYPE
   MAXIMUM COLOR INDEX, COLOR INDEX PRECISION
   COLOR PRECISION, INDEX PRECISION
   INTEGER/REAL PRECISION
   METAFILE ELEMENT LIST, DEFAULTS REPLACEMENT
   FONT LIST, CHARACTER SET LIST, CHARACTER CODING ANNOUNCER

3. Les descripteurs d'images (*Picture Descriptor*) :

   COLOR SELECTION MODE, SCALING MODE
   LINE WIDTH SPECIFICATION MODE
   MARKER SIZE SPECIFICATION MODE
   PERIMETER WIDTH SPECIFICATION MODE
   VDC EXTENT

4. Les instructions de contrôle :

   VDC INTEGER/REAL PRECISION, AUXILIARY COLOR
   CLIP RECTANGLE, CLIP INDICATOR

5. Primitives graphiques :

   CELL ARRAY
   CIRCLE, CIRCULAR ARC, CIRCULAR ARC CLOSE
   ELLIPSE, ELLIPTICAL ARC, ELLIPTICAL ARC CLOSE
   GDP
   POLYGON, POLYGON SET
   POLYLINE, DISJOINT POLYLINE
   POLYMARKER
   TEXT, APPEND TEXT, RESTRICTED TEXT

6. Attributs :

   LINE BUNDLE INDEX, LINE TYPE, LINE WIDTH, LINE COLOR
   MARKER BUNDLE INDEX, MARKER TYPE, MARKER SIZE, MARKER COLOR
   TEXT BUNDLE INDEX, TEXT FONT INDEX, TEXT PRECISION, TEXT COLOR
   CHARACTER EXPANSION FACTOR, CHARACTER SPACING, CHARACTER HEIGHT
   CHARACTER ORIENTATION, TEXT PATH, CHARACTER SET INDEX
   ALTERNATE CHARACTER SET INDEX, FILL BUNDLE INDEX
   INTERIOR STYLE, FILL COLOR, HATCH INDEX, PATTERN INDEX
   PERIMETER TYPE, PERIMETER WIDTH, PERIMETER COLOR, PERIMETER VISIBILITY
   FILL REFERENCE POINT, PATTERN SIZE, PATTERN TABLE, COLOR TABLE
   ASPECT SOURCE FLAGS

7. Echappement (pour implantation de commandes ou paramètres non standards) :

   ESCAPE

8. Données externes (*external*) :

   APPLICATION DATA, MESSAGE

Annexe C

Bibliographie

[1] E. Ahronovitz,
Algorithmes de compression d'images et codes de contours,
Thèse, Université de Saint-Etienne, septembre 1985.

[2] M. Alami, L. Peralta, L. Henninger and A. Osorio,
Coding picture edges by circle approach,
Proceedings of International Electronic Image Week, Second Image Symposium, Nice, pp. 180–184, 1986.

[3] C. Arcelli and G. Sanniti di Baja,
A width-independent fast thinning algorithm,
IEEE Trans. on PAMI, Vol. 7 No. 4 pp. 463–474, 1985.

[4] H. Asada and M. Brady,
The Curvature Primal Sketch,
IEEE Trans. on PAMI, Vol. 8 No. 1 pp. 2–14, 1986.

[5] N. Ayache and O.D. Faugeras,
HYPER: a new approach for the recognition and positioning of two-dimensional objects,
IEEE Trans. on PAMI, Vol. 8 No. 1 pp. 44–54, 1986.

[6] G. Bauer,
Following lines in vector data using smoothness or straightness criteria and applications,
Proceedings of International seminar on Symbol Recognition, Oslo, pp. 9–15, 1985.

[7] A. Belaïd et G. Masini,
Segmentation de tracés sur tablette graphique en vue de leur reconnaissance,
Techniques et Sciences de l'Informatique, Vol. 1 No. 2 pp. 155–168, 1982.

[8] M. Berthod et P. Jancenne,
Le pré-traitement des tracés manuscrits sur une tablette graphique,
Actes du 2ème Congrès AFCET de Reconnaissance des Formes et Intelligence Artificielle, pp. 195–209, 1979.

[9] R.E. Blake,
Some matching methods for symbol recognition,
Proceedings of International Seminar on Symbol Recognition, Oslo, 1985.

[10] P.R. Bono,
A Survey of Graphics Standards and Their Role in Information Interchange,
Computer Magazine, Vol. 18 No. 10 pp. 63–75, 1985.

[11] K. Braten, E. Holbaek-Hanssen and T. Taxt,
A general software system for supervised statistical classification of symbols,
Proceedings of International Seminar on Symbol Recognition, Oslo, 1985.

[12] R.L.T. Cederberg,
Chain-link code and segmentation for raster scan devices,
Computer Graphics and Image Processing, Vol. 10 No. 3 pp. 224−234, 1979.

[13] D. Colnet, G. Masini, A. Napoli, Y. Noiret et K. Tombre,
Les langages Orientés Objets,
rapport interne; CRIN 86−R−077, Centre de Recherche en Informatique de Nancy, 1986.

[14] U. Cugini, G. Ferri, P. Mussio, P. Protti,
Pattern-directed restoration and vectorization of digitized engineering drawings,
Computers and Graphics, Vol. 8 No. 4 pp. 337−350, 1984.

[15] W. Doster,
Different states of a document's content on its way from the Gutenbergian world to the electronic world,
Proceedings of Seventh International Conference on Pattern Recognition, Montreal, pp. 872−874, 1984.

[16] J.G. Dunham,
Optimum Uniform Piecewise Linear Approximation of Planar Curves,
IEEE Trans. On PAMI, Vol. 8 No. 1 pp. 67−75, 1986.

[17] I.D. Faux and M.J. Pratt,
Computational Geometry for Design and Manufacture,
Ellis Horwood Limited, pp. 308−310, 1979.

[18] H. Freeman,
Computer Processing of Line-drawing Images,
Computing Surveys, Vol. 6 No. 1 1974.

[19] F.C.A. Groen and R.J van Munster,
Topology based analysis of schematic diagrams,
Proceedings of Seventh International Conference on Pattern Recognition, Montreal, pp. 1310−1312, 1984.

[20] F.C.A. Groen, A.C. Sanderson and J.F. Schlag,
Symbol recognition in electrical diagrams using probabilistic graph matching,
Pattern Recognition Letters, Vol. 3 pp. 343−350, 1985.

[21] J.F. Harris, J. Kittler, B. Llwellyn, G. Preston,
Pattern Recognition Theory and Applications : A modular system for interpreting binary pixel representations of line-structured data,
D. Reidel Publishing Company, pp. 311−351, 1982.

[22] M. Hase and Y. Hoshino,
Segmentation Method of Document Images by Two-Dimensional Fourier Transform,
Systems and Computers in Japan, Vol. 16 No. 3 pp. 38−47, 1985.

[23] W, Horak,
Office Document Architecture and Office Document Interchange Formats: Current Status of International Standards,
Computer Magazine, Vol. 18 No. 10 pp. 50−60, 1985.

[24] D.A. Huffman,
A Method for the Construction of Minimum-Redundancy Codes,
Proceedings of the I.R.E., Vol. 40 pp. 1098−1101, 1952.

[25] H. Jansen and F-L Krause,
Interpretation of freehand drawings for mechanical design processes,
Computers and Graphics, Vol. 8 No. 4 pp. 351−369, 1984.

[26] E. Kawaguchi and T. Endo,
On a method of binary picture representation and its application to data compression,
IEEE Trans. On PAMI, Vol. 2 No. 1 pp. 27−35, 1980.

[27] H. Kida, O. Iwaki and K. Kawada,
Document recognition system for office automation,
Proceedings of Eighth International Conference on Pattern Recognition, Paris, pp. 446−448, 1986.

[28] G. Krönert, G. Lauber, J. Lörscher, E. Meynieux, W. Postl, U. Schneider, S. Seisen, K. Tombre,
Document editing and entry based on the standardised office document architecture,
Proceedings of ESPRIT Technical Week, Bruxelles, 1986.

[29] M. Kunt et O. Johnsen,
Codes de Blocs à Préfixe pour la réduction de la Redondance d'Images,
Annales des Télécommunications, Vol. 33 No. 7−8 1978.

[30] M.S. Landy, Y. Cohen,
Vectorgraph coding; efficient coding of line drawings,
Computer Vision, Graphics and Image Processing, Vol. 30 pp. 331−344, 1985.

[31] G. Masini and R. Mohr,
MIRABELLE, a system for structural analysis of drawings,
Pattern Recognition, Vol. 16 No. 4 pp. 363−372, 1983.

[32] L. Massip-Pailhes et R. Payrissat,
Codage et protection des informations dans un système de transmission d'image fonctionnant en temps réel,
Actes du premier Colloque Image CESTA, Biarritz, pp. 271−275, 1985.

[33] E. Meynieux, S. Seisen, K. Tombre,
Bilevel information recognition and coding in office paper documents,
Proceedings of Eighth International Conference on Pattern Recognition, Paris, pp. 442−445, 1986.

[34] E. Meynieux, W. Postl, S. Seisen, K. Tombre,
Office paper document analysis,
Proceedings of IFIP working conference on methods and tools for office systems, Pisa, 1986.

[35] R. Mohr,
Syntactical and structural pattern analysis, H. Bunke ed. : A general purpose line drawing analysis system,
Springer Verlag, 1987.

[36] F. Mokhtarian and A. Mackworth,
Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes,
IEEE Trans. on PAMI, Vol. 8 No. 1 pp. 34−43, 1986.

[37] H. Murase and T. Wakahara,
Online hand-sketched figure recognition,
Pattern Recognition, Vol. 19 No. 2 pp. 147−160, 1986.

[38] M. Nadler,
A survey of document segmentation and coding techniques,
Computer Vision, Graphics and Image Processing, Vol. 28 pp. 240−262, 1984.

[39] G. Nagy and S. Seth,
Hierarchical representation of optically scanned documents,
Proceedings of Seventh International Conference on Pattern Recognition, Montreal, pp. 347−349, 1984.

[40] G. Nagy, S. Seth and S.D. Stoddard,
Document analysis with an expert system,
Proceedings of Pattern Recognition in Practice II, Amsterdam, 1985.

[41] W. Postl,
Halftone Recognition by an Experimental Text and Facsimile Workstation,
Proceedings of Sixth International Conference on Pattern Recognition, Munich, pp. 489−491, 1982.

[42] W. Postl,
Detection of linear oblique structures and skew scan in digitized documents,
Proceedings of Eighth International Conference on Pattern Recognition, Paris, pp. 687−689, 1986.

[43] W.K. Pratt,
Digital Image Processing,
Wiley-Interscience, 1978.

[44] T. Pun,
Signal Processing : A new method for grey-level picture thresholding using the entropy of the histogram,
North-Holland Publishing Company, pp. 323−237, 1980.

[45] U. Ramer,
An iterative procedure for the polygonal approximation of plane curves,
Computer Graphics and Image Processing, Vol. 1 pp. 244−256, 1972.

[46] A. Rosenfeld,
Axial representation of shape,
Computer Vision, Graphics and Image Processing, Vol. 33 No. 2 pp. 156−173, 1986.

[47] A. Rosenfeld and J.L Pfaltz,
Sequential operations in digital pictures processing,
Journal of the Association of Computing Machinery, Vol. 13 No. 4 pp. 471−494, 1966.

[48] J. Serra,
Image Analysis and Mathematical Morphology,
Academic Press, 1982.

[49] S.N. Srihari and G.W. Zack,
Document image analysis,
Proceedings of Eighth International Conference on Pattern Recognition, Paris, pp. 434−436, 1986.

[50] O. Sylvère,
Vers une transformation d'un document sur support papier en document électronqiue : Méthode de délimitation et d'identification des zones informatives d'un document illustré d'origine papier,
Thèse, Université Pierre et Marie Curie, Paris 6, octobre 1986.

[51] K. Tombre,
Essai de classification des algorithmes de traitement d'images,
rapport de DEA; CRIN 83−R−057, Université de Nancy I, 1983.

[52] K. Tombre,
Segmentation and line extraction for recognition and coding of graphical parts in a document,
Proceedings of International Seminar on Symbol Recognition, Oslo, 1985.

[53] K. Tombre,
Finding and coding graphics in a composite document,
Proceedings of Fifth Scandinavian Conference on Image Analysis, Stockholm, 1987.

[54] J.P. Trincklin,
Conception d'un système d'analyse de documents. Etude et réalisation d'un module d'extraction de la structure physique de documents à support visuel,
Thèse, Université de Franche-Comté, Besançon, septembre 1984.

[55] K. Wall,
Curve fitting based on polygonal approximation,
Proceedings of Eighth International Conference on Pattern Recognition, Paris, pp. 1273−1275, 1986.

[56] K. Wall and P. Danielsson,
A fast sequential method for polygonal approximation of digitized curves,
Computer Vision, Graphics and Image Processing, Vol. 28 pp. 220−227, 1984.

[57] L.T. Watson, K. Arvind, R.W. Ehrich and R.M. Haralick,
Extraction of lines and regions from grey tone line drawing images,
Pattern Recognition, Vol. 17 No. 5 pp. 493−507, 1984.

[58] J.S. Weszka and A. Rosenfeld,
Histogram modification for threshold selection,
IEEE Trans. on Systems, Man and Cybernetics, Vol. 9 No. 1 pp. 38−52, 1979.

[59] J.M. White and G.D. Rohrer,
Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction,
IBM Journal of Research and Development, Vol. 27 No. 4 pp. 400−411, 1983.

[60] K.Y. Wong, R.G. Casey and F.M. Wahl,
Document Analysis System,
IBM Journal of Research and Development, Vol. 26 No. 6 pp. 647−656, 1982.

Table des Matières

Résumé

La bureautique prend actuellement un essor considérable, et de plus en plus de documents sont créés, manipulés et stockés sous forme électronique. Cependant, les documents sur support papier continuent à exister. Il est alors très utile de pouvoir saisir un document par l'intermédiaire d'une caméra ou d'un scanner, et de transformer l'image obtenue en un document électronique conforme aux normes existantes.

Dans cette thèse, nous présentons le résultat de notre travail sur ce problème, dans le cadre du projet ESPRIT **HERODE**. Plus précisément, nous nous intéressons à la saisie et à la reconnaissance de l'information graphique.

Après avoir énuméré les différentes étapes nécessaires, ainsi que les méthodes existantes, nous présentons la méthode originale que nous avons développée. Partant d'une image binaire, elle construit, par enrichissements successifs au cours des étapes du traitement, une structure de données qui nous permet de reconnaître des primitives graphiques pouvant être manipulées par un éditeur graphique.

Nous montrons pour terminer que ce travail ouvre des perspectives vers l'interprétation et la lecture intelligente de plans techniques.