Analyse et intégration de graphes de voisinage dans la méthode DMOS

Rapport de stage de master recherche

Aurélie LEMAITRE

Encadrée par Bertrand COUASNON et Ivan LEPLUMEY
Équipe IMADOC

Juin 2005

Remerciements

Je tiens tout d'abord à remercier mes encadreurs Bertrand Coüasnon, Chargé de recherche à l'INRIA en détachement et Ivan Leplumey, Maître de conférence à l'INSA de Rennes, pour leur disponibilité et leurs conseils avisés tout au long du stage.

Je remercie également Jean Camillerapp, Professeur à l'INSA de Rennes, pour ses explications sur la bibliothèque de vision précoce et la mise en place de fonctionnalités qui ont facilité mon travail.

Je remercie enfin le personnel de l'IRISA pour son amabilité et surtout toute l'équipe IMADOC pour son soutien et son accueil chaleureux.

TABLE DES MATIÈRES

Introduction

Dans le domaine de la reconnaissance optique des documents fortement structurés (partitions musicales, formules mathématiques, formulaires, tableaux...), le projet IMADOC a développé la méthode DMOS (Description et MOdification de la Segmentation) constituée d'un formalisme grammatical de position (EPF) et d'un analyseur associé. Le système permet ainsi de produire un analyseur du type de documents à traiter à partir d'une description grammaticale. Cette description bidimensionnelle intègre des opérateurs de position s'appuyant sur les positions relatives des boîtes englobantes des différents objets de l'image. Ces opérateurs manquent dans certains cas de précision.

L'objectif du stage est de réaliser d'autres opérateurs qui apporteront beaucoup plus de précision quant aux positions relatives d'objets voisins, et d'intégrer ces opérateurs à la méthode DMOS. Ces nouveaux opérateurs s'appuient sur l'utilisation d'un graphe de voisinage basé sur le pavage de Voronoï et les distances discrètes du chanfrein entre points contours d'objets proches. Ces opérateurs sont testés, entre autres, par la redéfinition d'une grammaire décrivant des registres manuscrits de décrets de naturalisation de la fin du 19ème siècle.

Dans un premier temps, nous allons donc voir quelques rappels bibliographiques sur la méthode DMOS, ainsi que sur la notion de voisinage, et plus particulièrement les pavages de Voronoï. Nous présenterons ensuite le travail réalisé pour la mise en oeuvre d'un graphe de voisinage et son intégration dans la méthode DMOS. Nous étudierons enfin le contexte applicatif de ces travaux.

CHAPITRE

1

Méthode générique de reconnaissance de documents : DMOS

Le projet IMADOC a développé une méthode de reconnaissance de la structure de documents fortement structurés, DMOS(Description et MOdification de la Segmentation). Cette méthode a déjà permis de reconnaître, entre autres, des partitions musicales, des tableaux, des formules mathématiques, ou des documents d'archives.

Dans un premier temps, nous proposons quelques rappels bibliographiques sur la reconnaissance de documents et le système DMOS. Nous détaillons ensuite le formalisme grammatical EPF (*Enhanced Position Formalism*) sur lequel se base la méthode DMOS.

1.1 Reconnaissance de documents

L'objectif de l'analyse et de la reconnaissance de documents est de convertir l'information présentée sur un support papier sous une forme symbolique équivalente, réutilisable par ordinateur. Kasturi *et al.* présentent dans [9] les différentes étapes nécessaires lors de l'analyse d'image. Ces étapes sont regroupées sur le schéma 1.1, proposé dans [6].

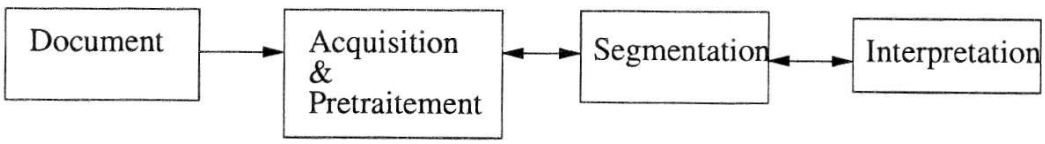


FIG. 1.1 – Schéma de principe de la reconnaissance de documents [6]

Acquisition L'acquisition du document initial est réalisée grâce à un capteur de type scanner. L'image subit alors des pré-traitements, tels que l'élimination du bruit. Si on souhaite travailler sur une image en noir et blanc, on passe par une phase de binarisation. Cette étape consiste à seuiller l'image pour séparer les objets, en noir, du fond blanc. De nombreux algorithmes présentés dans [9] permettent d'effectuer ces traitements. Cependant, de plus en plus de travaux s'appuient sur des images en niveaux de gris.

Segmentation L'étape de segmentation consiste à isoler des entités de l'image, analysables individuellement. Cette étape se base souvent sur la reconnaissance des composantes connexes, c'est à dire les ensembles de points noirs dans une image qui se « touchent ».

Mao *et al.* dans [17] présentent les différentes méthodes de segmentation utilisées classiquement et distinguent trois familles :

- les algorithmes descendants qui partent du document global et segmentent itérativement l'image en plus petites zone ;
- les algorithmes ascendants qui partent de chaque pixel et procèdent par regroupement de composantes connexes ;
- les algorithmes hybrides qui mixent les deux approches précédentes.

Cette étape de segmentation est particulièrement délicate : bien segmenter, c'est savoir séparer correctement le fond, de la forme à reconnaître. Pour cela, il faut donc avoir reconnu la forme. Inversement, le succès de la phase de reconnaissance s'appuie sur le bon fractionnement. C'est là le paradoxe de la segmentation.

Les problèmes classiques liés à la segmentation sont les suivants :

- la sur-segmentation, lorsque les formes sont trop fractionnées ;
- la sous-segmentation, lorsque des formes se « touchant » auraient du être séparées.

Interprétation Enfin, l'étape d'interprétation consiste à attribuer un sens à chacun des éléments segmentés, puis, par regroupement, à la totalité du document. De nombreux algorithmes de reconnaissance de caractères existent, mais pour pouvoir les appliquer, il faut connaître la nature et la structure des éléments à analyser.

Les connaissances acquises lors de cette étape d'interprétation peuvent, si nécessaire, être utilisées pour reprendre la segmentation et l'améliorer.

1.2 Système DMOS

1.2.1 Principe global

Tout système de reconnaissance de documents nécessite l'introduction de connaissances liées au type de document à reconnaître. Dans la plupart de ces systèmes, la connaissance est noyée dans le code et, de ce fait, difficile à adapter à de nouveaux types de documents.

Le système DMOS (Description et MOdification de la Segmentation), conçu au sein du projet IMADOC, s'appuie sur une séparation de la connaissance liée au document, afin de pouvoir reconnaître des documents structurés de types variés.

Dans [7] et [5], B. Coüasnon présente les points clés de la méthode :

- le formalisme grammatical EPF (*Enhanced Position Formalism*), langage de description pour les documents structurés, pouvant représenter à la fois des notions graphiques, syntaxiques et sémantiques ;
- l'analyseur associé capable de modifier en cours de reconnaissance la structure à analyser, en faisant appel à une modification de la segmentation selon le contexte ;
- le détecteur de terminaux que sont les segments de droites et les symboles.

DMOS est une méthode générique puisque son adaptation à un nouveau type de documents est réalisée simplement par la définition d'une nouvelle grammaire, dans le langage EPF. Ainsi, cette méthode a été validée face à de nombreux problèmes tels que la reconnaissance de formules mathématiques, de partitions musicales, de documents d'archives (dans [4]) ou la détection de courts de tennis (dans [5]).

1.2.2 Architecture de reconnaissance

L'architecture du système est présentée figure 1.2 d'après la description faite dans [6].

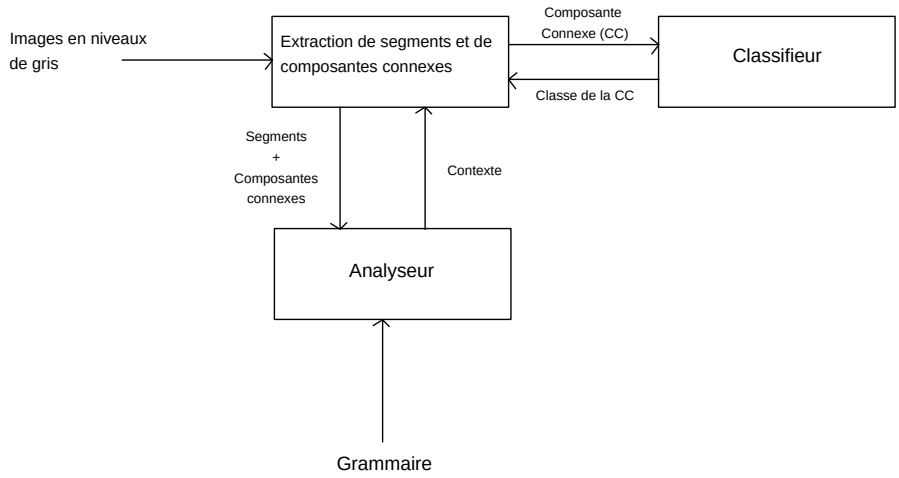


FIG. 1.2 – Architecture du système DMOS

A partir d'une image en niveaux de gris, les segments et les composantes connexes sont extraites. Ces composantes connexes sont envoyées vers le classifieur qui va leur associer, si possible, une classe d'appartenance (un 'a', un '1',...). Si la segmentation est incorrecte, ou si la composante n'appartient pas au vocabulaire du classifieur, elle est rejetée. L'analyseur, généré grâce à la définition de la grammaire au format EPF, dispose donc en entrée de la liste des segments et de celle des composantes, éventuellement étiquetées. Il va effectuer la reconnaissance du document.

Au cours de l'analyse, l'analyseur peut demander une modification de la segmentation de l'image, et donc de la structure analysée. En effet, le contexte modélisé par la grammaire permet de faire des hypothèses de segmentation qui peuvent être validées par le classifieur. Une autre manière de remettre en cause la segmentation consiste à supprimer les segments déjà reconnus, puis à demander une nouvelle segmentation sur les éléments restants, qui va transformer des composantes connexes en des composantes bien segmentées.

1.3 Formalisme EPF

La description de chaque catégorie de documents à analyser passe par la définition d'une grammaire. Nous présentons ce concept au travers d'un exemple, avant de définir plus formellement les éléments du langage. Nous étudions ensuite les mécanismes de voisinage utilisés pour l'analyse.

1.3.1 Principe du langage

La méthode DMOS est basée sur le formalisme grammatical EPF.

Les terminaux d'un document peuvent être de deux types :

– les segments, à tendance horizontale ou verticale ;
– les composantes identifiées par le classifieur.

Dès qu'un terminal est reconnu et consommé, il est retiré de la structure analysée.

Outre les non-terminaux qu'on rencontre classiquement dans une grammaire, le formalisme EPF contient des opérateurs de position permettant une analyse en deux dimensions. Ces opérateurs permettent d'exprimer l'organisation bidimensionnelle des documents en précisant la position relative ou absolue des différents éléments à reconnaître.

Ces opérateurs sont à la base de la méthode d'analyse. En effet, habituellement, lorsqu'on applique une grammaire pour une analyse, on ne se pose pas la question du prochain terminal à reconnaître : les terminaux s'enchaînent sous la tête de lecture (cas d'une chaîne de caractères, par exemple). Dans le cas du document en deux dimensions, il faut savoir à quel endroit aller chercher le prochain terminal à reconnaître. Les opérateurs de positions ont donc pour rôle d'indiquer où trouver le prochain symbole.

Les règles de grammaire sont donc composées de terminaux et d'opérateurs de position, ainsi que de non-terminaux structurant le document. Elles contiennent également d'autres opérateurs que nous allons détailler par la suite.

1.3.2 Exemple de grammaire

Nous introduisons d'abord les principaux opérateurs de manière intuitive, sur un exemple, avant de donner une description plus formelle du langage.

L'exemple de grammaire présenté ici a pour but de décrire simplement une formule d'intégrale mathématique, telle que celle présentée figure 1.3.

Fig. 1.3 – Formule d'intégrale

Intuitivement, on peut dire qu'une telle formule se décompose en :
- un symbole d'intégrale, situé dans la partie gauche de l'image ;
- des bornes d'intégration situées sur les parties basses et hautes de l'intégrale. On considère ici qu'elles sont constituées chacune d'un seul élément ;
- une expression, située à droite de l'intégrale, composée d'une suite de composantes alignées.

La règle de grammaire principale va donc décrire cela. Pour traduire les notions de positionnement, on utilise les opérateurs de position :
- `AT` qui permet de préciser un positionnement relatif par rapport à un objet précédemment trouvé ;
- `AT_ABS` qui permet de préciser un positionnement absolu par rapport au document global.

Les différents éléments d'une règle sont concaténés par l'opérateur `&&`. L'opérateur `##` est un opérateur de factorisation, qui permet ici décrire chacune des trois dernières lignes de la règle en se basant sur la première.

La règle principale se décrit donc ainsi :

```
formuleIntegrale::=
    AT_ABS(gaucheImage) && signeIntegrale && (
        AT(hautDroite signeIntegrale) && borneSup ##
```

```
        AT(basDroite signeIntegrale) && borneInf ##
        AT(droite signeIntegrale) && expression).
```

Les règles `signeIntegrale`, `borneSup` et `borneInf` correspondent à la détection de terminaux. La règle `expression` est un non terminal qui permet de détecter une succession de caractères. Il s'agit d'une règle récursive qui peut être codée de la manière suivante :

```
expression::= caractere.
expression::= caractere &&
    AT(droite caractere) && expression.
```

Cela décrit le fait qu'une expression est une suite de caractères (au moins un) situés à la droite les uns des autres.

1.3.3 Les opérateurs du langage

Nous allons voir maintenant plus formellement les opérateurs du langage.

1.3.3.1 Les opérateurs *AT* et *AT_ABS*

La syntaxe de ces opérateurs de position est la suivante :

```
AT(position)
AT_ABS(position)
```

L'opérateur `AT` est utilisé pour un positionnement relatif, tandis que `AT_ABS` est utilisé pour un positionnement absolu. Un cas d'utilisation classique est le suivant :

```
A && AT(pos) && B
```

qui se traduit par : partant de `A`, le (non-)terminal de référence, on va chercher `B` dans la *zone* définie par `pos`. Cette zone est un polygone. L'opérateur `&&` utilisé ici est l'opérateur de concaténation. Les éléments `A` et `B` sont des terminaux ou des non-terminaux. On peut définir dans la grammaire autant de positionnements *pos* que nécessaire.

1.3.3.2 L'opérateur *IN DO*

Cet opérateur permet de réduire la zone d'application d'une règle de la grammaire. Il est formulé de la manière suivante :

```
IN(zone) DO (regle)
```

Cela signifie que la `regle` ne doit s'appliquer que dans la `zone`. Cet opérateur est très utile notamment pour faire des définitions récursives et limiter les recherches dans une sous partie du document : une case d'un tableau, par exemple.

1.3.3.3 Les opérateurs de reconnaissance des terminaux

Les opérateur `TERM_SEG` et `TERM_CMP` sont utilisés pour la reconnaissance, respectivement, des segments et des composantes connexes, dans la zone de recherche fixée. Leur syntaxe est la suivante :

```
TERM_CMP PreCondition PostCondition Etiquette ComposanteReconnue
TERM_SEG PreCondition PostCondition Etiquette SegmentReconnu
```

Lors de la recherche de la composante dans une zone, on peut vouloir ne pas prendre la première trouvée, mais celle respectant une certaine condition. Par exemple, on peut rechercher un symbole qui soit un chiffre ou un segment qui soit horizontal. Ce souhait est exprimé grâce à la `PreCondition`, définie à la demande. Lorsque la composante est trouvée, la `PostCondition` permet de vérifier un critère d'acceptation. L'`Etiquette` est le nom qui sera donné à l'élément reconnu.

1.3.3.4 Les opérateurs de sauvegarde

Les opérateurs `--->` et `<---` permettent de mémoriser des relations bi-directionnelles entre des composantes. Par exemple, dans la règle « on reconnaît A, on détecte B puis on reconnaît à nouveau A », on voit la nécessité de stocker le premier A pour pouvoir le retrouver. La syntaxe est la suivante :

```
(A ---> labelDeA) && AT(pos1) && B && AT(pos2) && (A <--- labelDeA)
```

Ici, on sauvegarde `A` sous le nom `labelDeA` pour pouvoir le ré-utiliser par la suite. Les étapes sont :

- on reconnaît `A` et on mémorise son identité dans `labelDeA` ;
- partant de `A`, on va chercher dans la zone `pos1` ;
- on reconnaît `B` ;
- partant de `B`, on va chercher dans la zone `pos2` ;
- on reconnaît le même `A` que celui mémorisé auparavant.

Afin de gérer la portée des libellés utilisés, on dispose de l'opérateur suivant :

```
DECLARE(reference) (ensemble_de_regles_ou_reference_est_valide)
```

1.3.3.5 L'opérateur *FIND UNTIL*

Cet opérateur a pour but de permettre la gestion de bruit dans les documents.

Dans l'analyse classique, pour traiter un élément d'une image (composante ou segment), on essaie toutes les règles jusqu'à ce que l'une s'applique. On échoue sinon. L'opérateur `FIND UNTIL` permet de faire le contraire : on va essayer une règle sur tous les éléments de la structure à analyser. Sa syntaxe est la suivante :

```
FIND(Règle) UNTIL(ConditionArret)
```

Cet opérateur permet de boucler sur la règle `Règle` jusqu'à la condition d'arrêt `ConditionArret`. Ainsi, cet opérateur permet d'essayer la règle sur toutes les composantes ou tous les segments jusqu'à ce qu'elle réussisse ou que la condition d'arrêt soit vérifiée. Si cette condition d'arrêt réussit, l'opérateur échoue.

1.3.3.6 L'opérateur *SELECT*

Cet opérateur permet d'appliquer une règle sans consommer les terminaux rencontrés. Sa syntaxe est :

```
SELECT(Règle)
```

`SELECT` peut ainsi permettre de détecter une structure globale de document avant de faire la reconnaissance explicite des terminaux.

1.3.4 Mécanismes d'analyse des terminaux

Nous avons vu que l'analyse des terminaux passe par l'utilisation conjointe d'un opérateur de position `AT` ou `AT_ABS`, et d'un détecteur de terminaux `TERM_SEG` ou `TERM_CMP`. Nous détaillons ici le mécanisme de détection bidimensionnelle induit par ces opérateurs, avant de voir leur limite.

1.3.4.1 Détection bidimensionnelle

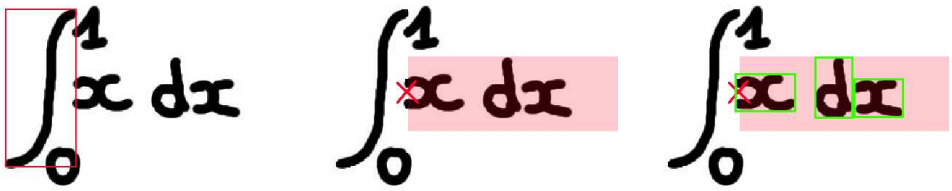
L'analyse des terminaux s'appuie sur deux outils :
- une zone de recherche ;
- un point d'ancrage.

On distingue donc deux étapes :
- les opérateurs de position permettent de fixer un point d'ancrage, dépendant de l'objet de référence précédent, ainsi qu'une zone de recherche : rectangle ou polygone ;
- les opérateurs `TERM_SEG` et `TERM_CMP` procèdent alors au choix d'un terminal : dans cette zone de recherche, celui vérifiant les pré-conditions et étant le plus proche en distance du point d'ancrage.

Par exemple, sur la grammaire de l'intégrale, on s'intéresse aux instructions suivantes qui visent à extraire le premier caractère de l'expression de l'intégrale :

```
AT(droite signeIntegrale) &&
TERM_COMP noCond noCond caractere MonCaractere.
```

On suppose que l'on vient de reconnaître le `signeIntegrale`, caractérisé par sa boîte englobante sur la figure 1.4(a). L'exécution de l'instruction `AT` a pour effet de fixer le point



(a) Boîte englobante de l'intégrale

(b) Définition de la zone de recherche et du point d'ancrage

(c) Recherche des composantes selon leurs boîtes englobantes

FIG. 1.4 – Mécanisme de détection du x depuis le *signeIntegrale*

d'ancrage et la zone de recherche, à la position `droite`, par rapport à la boîte englobante du `signeIntegrale` (figure 1.4(b)). L'appel de l'instruction `TERM_COMP` va permettre de rechercher, dans la zone en rouge, le `caractere` dont la boîte englobante (figure 1.4(c)) est la plus proche du point d'ancrage. Aucune condition n'est nécessaire ici pour le terminal (`noCond`). Le terminal détecté est donc le x.

1.3.4.2 Limite des opérateurs de position

La distance qui est prise en compte ici est la distance euclidienne entre les boîtes englobantes des éléments et le point d'ancrage. Cette approximation des objets à un rectangle

est souvent suffisante. Cependant, dans certains cas, on aimerait pouvoir disposer d'une notion de positionnement relatif plus fine des objets.

DMOS, qui est une méthode générique, doit savoir analyser des configurations plus complexe, et dans la description de la grammaire EPF pour certains types de document, on se rend compte que l'approximation à des boîtes englobantes est un facteur limitant.

Par exemple, dans la grammaire présentée précédemment, la zone de recherche du x est définie par rapport à la boîte englobante de l'intégrale. Cette zone est incorrecte s'il y a un fort recouvrement entre les boîtes englobantes du x et de l'intégrale, comme sur la figure 1.5.

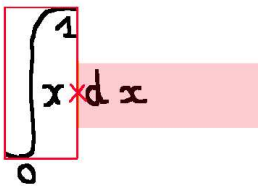


FIG. 1.5 – Zone de recherche du x erronée

On cherche donc à exprimer la notion de voisinage de manière plus précise qu'une approximation à des boîtes englobantes.

CHAPITRE

2

Notion de voisinage et pavage de Voronoï

Nous avons vu dans la section précédente l'importance de la notion de positionnement entre objets pour la reconnaissance de documents structurés. Nous présentons quelques rappels bibliographiques sur les notions de voisinage d'une manière générale, avant de nous intéresser plus spécifiquement aux diagrammes de Voronoï et à leurs applications en analyse de documents.

2.1 Notions de voisinage

Nous rappelons ici quelques notions utilisées en analyse d'image pour la modélisation du voisinage.

2.1.1 Voisinage d'un point

Une image binaire est codée par une matrice d'entiers. Un point p de l'image est défini par ses coordonnées cartésiennes (abscisse, ordonnée). Le point p a 4 voisins directs et 4 voisins indirects (figure 2.1).On définit donc le $4-$*voisinage* et le $8-$*voisinage* selon qu'on considère uniquement les voisins directs ou à la fois les voisins directs et indirects du point.

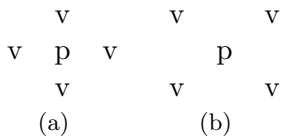


FIG. 2.1 – Voisins directs de p (a) et indirects (b)

2.1.2 Voisinage entre composantes

Burge *et al.* présentent en 1997 dans [3] une étude bibliographique en matière de *voisins* et de *voisinage*. Ils classifient les algorithmes de voisinage en deux catégories :

– les voisins ou voisinages *paramétrés* nécessitent la fixation d'une valeur de seuil : c'est le cas des k-plus-proches-voisins ou de la recherche dans un rayon fixé autour du point de référence ;

– les voisins ou voisinages *non paramétrés* ne nécessitent pas de paramètre spécifique. On compte par exemple les arbres de poids minimal, les graphes de Gabriel et les diagrammes d'aire de Voronoï.

Burge *et al.* exposent ensuite leur définition intuitive du voisinage d'un élément : il s'agit de la portion du plan euclidien qui est plus proche d'un élément que de n'importe quel autre. A partir de cette définition, ils remarquent qu'avec des algorithmes tels les graphes de Gabriel, une région du plan peut être voisine de plusieurs éléments à la fois. Ces algorithmes ne satisfont donc pas leur définition de voisinage.

Burge *et al.* insistent sur l'importance du *voisinage complet minimal de l'élément E* : c'est l'ensemble $N(E)$ des voisins de E,

– minimal au sens où seuls les voisins les plus proches de E sont inclus ;
– complet au sens où il contient tous les éléments qui sont à proximité.

Les algorithmes appartenant à la classe des voisinages paramétrés ne vérifient pas cette propriété, puisque selon la valeur de seuil choisie, on n'a pas dans le cas général minimalité et complétude.

On remarque que les diagrammes d'aire de Voronoï, qui à chaque point associent une région d'influence en matière de voisinage, respectent à la fois :

– le fait qu'il n'y ait pas de paramètre pour décrire le voisinage ;
– la définition intuitive du voisinage qu'on pourrait avoir, puisque chaque région du plan a un seul objet comme plus proche voisin ;
– les conditions de complétude et de minimalité du voisinage, puisqu'à chaque point, on associe exactement tous ses voisins.

Ce sont donc ces diagrammes que nous utilisons pour proposer une alternative à la notion de voisinage utilisée dans DMOS. Nous allons maintenant les présenter plus en détails.

2.2 Diagrammes de Voronoï

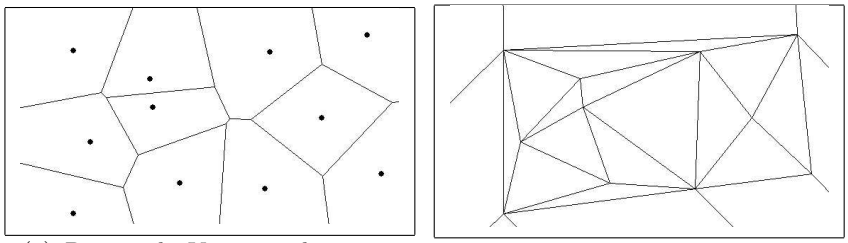
Les diagrammes de Voronoï sont souvent utilisés dans le domaine de l'analyse de documents, et plus précisément dans le cadre de la reconnaissance de la structure de documents. Après une présentation intuitive de ces diagrammes, nous donnerons une définition du diagramme ordinaire puis nous présenterons les possibilités de généralisation.

2.2.1 Présentation

Les diagrammes de Voronoï sont des diagrammes de voisinage. Appelés aussi mosaïques ou pavages, ils représentent un découpage de l'espace tel qu'autour de chaque point on regroupe toute la zone de l'espace plus proche de ce point que de n'importe quel autre. Un exemple est donné figure 2.2(a). On utilisera indifféremment la terminologie « diagramme » ou « pavage » de Voronoï. Le « graphe » de Voronoï représente plus précisément les frontières du pavage.

Au diagramme de Voronoï, on associe le graphe dual : le graphe de Delaunay. Ce graphe est composé des arêtes entre points ayant une frontière commune. Un exemple est donné figure 2.2(b).

Nous allons donner une définition plus formelle de ces diagrammes.



(a) Pavage de Voronoi : les voisinages des points sont délimités

(b) Diagramme de Delaunay : les points voisins sont reliés entre eux

FIG. 2.2 – Exemple de diagrammes

2.2.2 Diagramme ordinaire de Voronoï

2.2.2.1 Définitions

Okabe *et al.* présentent les définitions des diagrammes de Voronoï dans [18].

Définition : Région de Voronoï Soit P $=\{p_1, ..., p_n\}$ une ensemble de points du plan, appelés *germes*, et $d(p, q)$ la distance *euclidienne* entre les points p et q. La région de Voronoï d'un point p_i de P, est la région donnée par

$$V(p_i) = \{p | d(p, p_i) \leq d(p, p_j), \forall j \neq i\}$$

C'est l'ensemble des points plus proches de p_i que de tout autre point. On note $\delta V(p_i)$ l'ensemble des *frontières* de la région de Voronoï $V(p_i)$.

Définition : Diagramme de Voronoï Le diagramme de Voronoï ordinaire généré par l'ensemble des points P est donné comme l'ensemble des régions de Voronoï :

$$V(P) = \{V(p_i), ..., V(p_n)\}$$

Définition : Graphe de Delaunay (d'après [1]) Soit P $=\{p_1, ..., p_n\}$ une ensemble de points du plan, le graphe de Delaunay $Del(P)$ a pour sommets les points de P et pour arêtes les segments joignant les points p et q qui sont voisins, c'est à dire qui partagent une même frontière :

$$\delta V(p) \cap \delta V(q) \neq 0$$

2.2.2.2 Construction

De nombreux algorithmes ont été mis au point pour la construction des diagrammes de Voronoï basés sur des points, à la fois en 2D et en 3D. Ces algorithmes sont cités par Attali dans [1] et Bertin dans [2]. Une implémentation est proposée dans [18].

Kise *et al.* dans [11] synthétisent en disant que la construction du diagramme a une complexité en $O(n \log n)$, dans le pire des cas, mais en $O(n)$ en moyenne, où n est le nombre de germes.

2.2.3 Diagramme généralisé de Voronoï

2.2.3.1 Principe de la généralisation

Si les diagrammes de Voronoï sont très utilisés en analyse d'image, c'est qu'il est possible de généraliser la définition précédente. Okabe *et al.* proposent dans [18] de nombreuses méthodes de généralisation. Burge *et al.* les classifient dans [3] selon trois critères paramétrables :

- la mesure de distance : euclidienne, discrète ;
- l'espace : plan cartésien, 3D sphérique ;
- l'ensemble des germes : points, surfaces.

Dans le cas de l'analyse de documents, le diagramme de Voronoï généralisé qui est utilisé est celui dans le plan cartésien, généré à partir de surfaces (les composantes connexes). Nous allons en donner la définition formelle.

2.2.3.2 Diagramme de Voronoï basé sur des surfaces

Les définitions formelles sont données par Kise *et al.* dans [11].

Définition : Diagramme de Voronoï basé sur des aires Soit $G = \{g_1, ..., g_n\}$ un ensemble de figures ne se chevauchant pas, dans le plan et soit $d(p, q)$ la distance entre les points p et q. Alors la distance entre un point p et une figure g_i est définie par

$$d(p, g_i) = \min_{q \in g_i} d(p, q)$$

La région de Voronoï $V(g_i)$ et le diagramme de Voronoï basé sur les aires $V(G)$ sont définis par

$$V(g_i) = \{p | d(p, g_i) \leq d(p, g_j), \forall j \neq i\}$$

$$V(G) = \{V(g_1), ..., V(g_n)\}$$

Un exemple de diagramme basé sur les surfaces est présenté figure 2.3.

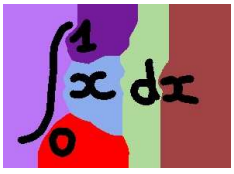


FIG. 2.3 – Exemple de diagramme de Voronoï basé sur les surfaces

2.2.4 Intérêts de Voronoï

Les diagrammes de Voronoï généralisés offrent des propriétés très intéressantes pour le traitement d'images :

- ils peuvent être construits indépendamment de la structure du document ou de l'inclinaison de l'image ;
- ils ne requièrent aucun paramètre lors de la construction (par opposition aux k-plus proches voisins, par exemple) ;
- ils satisfont des propriétés de voisinage telles que la minimalité, la complétude et la symétrie (voir section 2.1.2) ;

– dans le cas de la segmentation d'images, les frontières entres composantes sont déjà incluses dans le diagramme ;
– on passe très facilement du pavage de Voronoï au graphe de voisinage de Delaunay.

Cette étude confirme donc l'intérêt d'exploiter un diagramme de Voronoï pour analyser le voisinage.

2.3 Applications existantes

Les diagrammes de Voronoï sont utilisés pour la détection de structure de documents. Nous présentons ici les travaux de Lu *et al.* et Kise *et al.* qui ont travaillé respectivement sur l'extraction des mots et l'extraction de lignes et de paragraphes. Dans les deux cas, leurs travaux s'appuient sur un diagramme de Voronoï basé sur les aires et la distance euclidienne. Disposant du diagramme, ils appliquent une technique de sélection de frontières pour effectuer la segmentation.

2.3.1 Segmentation comme sélection de frontière

Le principe de base de la segmentation de documents en utilisant le diagramme de Voronoï est la sélection de frontières. En effet, chaque élément à reconnaître dans la page (mot, bloc de textes, zone d'image) va être formé d'un regroupement de composantes connexes, et donc de régions de Voronoï adjacentes. Le principe de la structuration de documents ou de l'extraction de mots est donc basé sur la sélection des frontières de Voronoï : lesquelles garder, lesquelles supprimer. Pour chacune des applications présentées, le problème va donc être de déterminer des critères de sélection des frontières de Voronoï.

2.3.2 Structuration de documents

Nous présentons ici les travaux de Kise *et al.* concernant la structuration d'images de documents. Ces travaux ont été présentés en 1997 dans [12], et approfondis dans [11], puis discutés en 1999 dans [10].

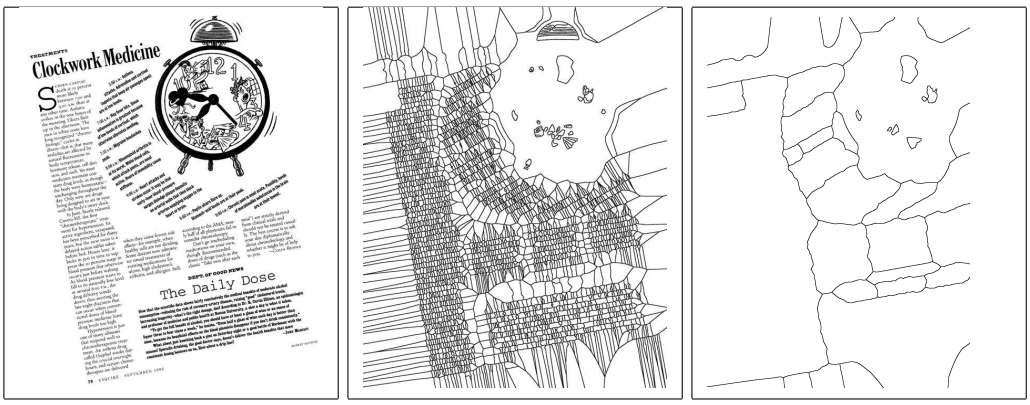
Le but de ces travaux est d'extraire les régions de texte dans des documents n'ayant pas nécessairement une structure classique : le texte n'est pas forcément disposé selon des rectangles et l'image peut avoir été acquise avec un certain biais (exemple de document figure 2.4(a)). En appliquant le concept présenté dans la section 2.3.1, le problème revient à construire le diagramme d'aire de Voronoï puis à trouver les bons critères de sélection d'arêtes à supprimer pour unifier les blocs de texte et les images.

Deux mesures sont utilisées pour la sélection :
– la distance minimale entre deux éléments $d(E)$;
– le rapport de surface entre deux éléments $ar(E)$.

En effet, en général, les blancs entre caractères, mots et lignes de texte sont plus faibles que les blancs entre colonnes. Les composantes très proches au vu de $d(E)$ pourront donc voir leur frontière supprimée. Cependant, les frontières entre du texte et des figures sont parfois faible ($d(E)$ petit). C'est pourquoi on a recours à un autre facteur : $ar(E)$. En effet, les composantes connexes des images ont une surface beaucoup plus importante que celles du texte ; le critère $ar(E)$ permet donc de les différencier.

A partir de ces mesures, des règles bornées par des seuils sont établies permettant de fixer quelles sont les frontières à éliminer. Ces seuils sont appris de manière statistique par comptage sur le document.

La figure 2.4 présente le type de résultats obtenus.



(a) Image originale (b) Diagramme de Voronoï basé sur les aires (c) Résultat de la segmentation

FIG. 2.4 – Résultats obtenus par Kise *et al.* [10]

2.3.3 Regroupement de mots

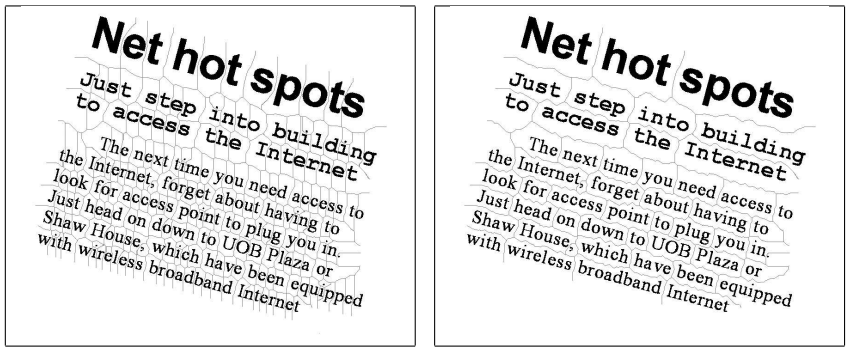
Lu *et al.* proposent des travaux sur le regroupement et l'extraction des mots dans un document : en 2003 dans [15] et en 2004 dans [16]. Leur raisonnement s'appuie également sur la logique présentée section 2.3.1. Supposant construit le diagramme d'aire de Voronoï, on cherche des critères pour éliminer les frontières à l'intérieur des mots.

Deux types de mesures sont appliquées :

– la distance minimale de la plus proche de deux composantes à la frontière entre elles $md_e(E)$;

– la distance minimale d'une composante à la plus proche de ses frontières $md_e(C)$.

A partir de ces valeurs, des formules sont exprimées en fonction de seuils, permettant de distinguer un espace entre deux lettres d'un espace entre deux mots où entre deux lignes, mais aussi les cas particuliers de la ponctuation. Comme dans la méthode précédente, les seuils sont calculés par un apprentissage statistique. Un exemple de résultat obtenu est présenté figure 2.5.



(a) Pavage de Voronoï (b) Regroupement des mots

FIG. 2.5 – Résultats obtenus par Wang *et al.* [16]

2.3.4 Remarques sur l'exploitation du diagramme

Dans le cadre des applications présentées, les documents étudiés sont relativement ciblés. Puisque les seuils sont appris sur l'intégralité des pages, les critères de détection fonctionnent mieux sur des documents où les régions de textes sont dominantes et uniformes (taille de police homogène, même interligne). Grâce à la régularité de l'espacement des caractères entre mots, entre lignes, entre paragraphes, on peut effectuer une segmentation sans tenir compte de l'orientation de la page.

Cette méthode qui convient bien pour des documents typographiés, semble donc plus délicate à mettre en oeuvre sur des documents manuscrits, où les espaces entre les mots ne sont pas réguliers, et encore moins entre les lignes. C'est donc la difficulté que nous avons rencontré dans notre exemple d'application qui cherche également à extraire des mots et des lignes d'un document.

La première étape de l'implémentation réalisée a été celle du diagramme de Voronoï basé sur les surfaces.

CHAPITRE

3

Mise en oeuvre d'un graphe de voisinage

Le graphe de voisinage que nous avons choisi d'utiliser se base sur la distance discrète du chanfrein. Nous proposons donc dans un premier temps quelques rappels bibliographiques sur les notions théoriques avant de présenter le travail de mise en oeuvre réalisé.

3.1 Distances discrètes et masque de chanfrein

Les distances discrètes permettent une approximation de la distance euclidienne réelle. Après une présentation globale, nous traiterons plus particulièrement des masques de chanfrein.

3.1.1 Distances discrètes

3.1.1.1 Motivations

La perception que nous avons des distances dans le monde réel est totalement basée sur la distance euclidienne. Dans le domaine du traitement d'images, ces modèles continus sont encore majoritairement utilisés. Cependant, dans la mesure où la nature même des images numériques est le modèle discret (ensemble de pixels), on peut se poser la question de l'intérêt du modèle continu. De plus, les coûts de calcul de la géométrie discrète sont souvent plus faibles que ceux des modèles continus, dans la mesure où toutes les valeurs restent dans le domaine des entiers. C'est pourquoi la géométrie discrète s'est développée dans le monde de l'analyse d'images.

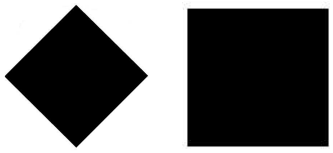
3.1.1.2 Définitions et propriétés

Définition : Distance discrète On appelle distance discrète sur un espace $\mathbb{E}$ une application $d : \mathbb{E} \times \mathbb{E} \longrightarrow \mathbb{N}$ vérifiant : $\forall A, B, C \in \mathbb{E}$

1. Définie positive : $d(A, B) \geq 0; d(A, B) = 0 \Leftrightarrow A = B$
2. Symétrie : $d(A, B) = d(B, A)$
3. Inégalité triangulaire : $d(A, B) \leq d(A, C) + d(C, B)$

Initialement, les distances discrètes utilisées sont d_4 et d_8, liées aux notions de $4-voisinage$ et de $8-voisinage$ présentées section 2.1.1, et ce grâce à leur facilité de calcul : chacun des voisins est placé à une distance de 1 (figure 3.1).

Parmi les distances utilisées, citons les distances de city block et de chessboard :



(a) (b)

FIG. 3.1 – Disques unité de (a) d_4 et (b) d_8

– city block : $d_4(A,B) = |x_b - x_a| + |y_b - y_a|$;
– chessboard : $d_8(A,B) = max(|x_b - x_a|; |y_b - y_a|)$.

Le but du jeu étant de se rapprocher au maximum de la distance euclidienne d_E, d'autres distances ont été envisagées, notamment $(d_E)^2$ ou la partie entière $int(d_E)$. Cependant, ces fonctions ne respectent pas l'inégalité triangulaire, surtout pour les petites valeurs. Une des solutions consiste à utiliser les masques de chanfrein.

3.1.2 Masques de chanfrein

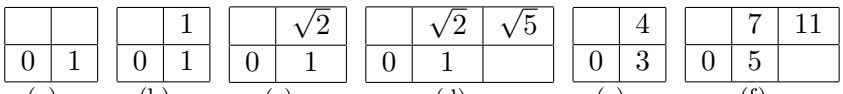
Thiel a largement étudié la géométrie des distances de chanfrein et présente les résultats de ses travaux dans [19] et [20].

Les distances de chanfrein sont des distances basées sur un masque de pondération dont l'application locale permet le calcul de la distance. Tous les déplacements autorisés à partir d'un point sont pondérés. On calcule la distance entre deux points comme le coût du chemin de coût minimal reliant les deux points en suivant les déplacements autorisés. Toute cette méthode est donc basée sur les pondérations choisies : le masque.

Dans la définition de d_4 et d_8, on associait un coût de 1 à chacun des déplacements directs ou diagonaux. Cela correspond donc aux masques présentés figures 3.2(a) et 3.2(b).

Idéalement, on cherche à approcher la distance euclidienne qui s'appliquerait par un masque $(1, \sqrt{2})$, présenté sur la figure 3.2(c). Ce masque peut être étendu pour avoir plus de précision au masque $(1, \sqrt{2}, \sqrt{5})$ présenté figure 3.2(d).

Thiel, dans [19] explique que les masques $(1, \sqrt{2})$ et $(1, \sqrt{2}, \sqrt{5})$ sont approximés respectivement par $(1, 4/3)$ et $(1, 7/5, 11/5)$ ce qui se traduit par les masques de chanfrein $(3, 4)$ et $(5, 7, 11)$ présentés figures 3.2(e) et 3.2(f).

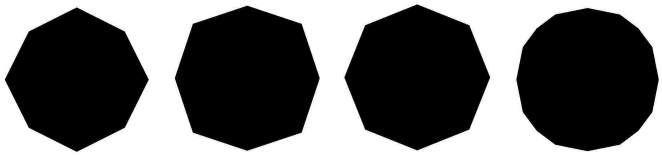


(a) (b) (c) (d) (e) (f)

FIG. 3.2 – Pondérations locales [19]

Le calcul global des distances s'effectue par propagation de distances locales. Ainsi, la figure 3.3 présentes les boules unité construites grâce à différents masques. On voit ici que plus le masque est large et les valeurs élevées et plus l'approximation de la distance euclidienne est bonne.

Il existe de nombreux algorithmes pour calculer des images de distance à partir des masques de chanfrein. Thiel étudie dans [20] les propriétés théoriques de masques de chanfrein et les applications mathématiques (construction de normes...).



(a) Masque (2,3) (b) Masque (3,4) (c) Masque (5,7) (d) Masque (5,7,11)

FIG. 3.3 – Disques unité de plusieurs masques de chanfrein

Parmi les applications possibles, il cite la construction d'un diagramme de Voronoï généralisé discret. Gribov *et al.* utilisent effectivement les distances discrètes pour la construction de leur diagramme dans [8]. C'est également sur ces distances que se base le graphe de voisinage que nous allons étudier.

3.2 Principe global du graphe de voisinage

Nous présentons maintenant le graphe de voisinage basé sur l'utilisation des distances discrètes proposé par Leplumey *et al.* dans [14].

3.2.1 Principe

Le but de ce graphe est d'établir des relations de voisinage entre les points contours de chacune des composantes connexes reconnues dans une image. Ce graphe s'appuie sur le choix d'une distance discrète : distance d_4 du « city block », distance d_8 du « chessboard », distance de chanfrein. Par application des masques et propagation des distances, on parvient à construire le pavage de Voronoï.

3.2.2 Construction du graphe

La construction du graphe de voisinage se déroule selon trois phases (Figure 3.4) :
- l'extraction des composantes connexes ;
- le calcul d'une image de distances ;
- l'extraction du graphe.

Extraction des composantes connexes : A partir d'une image initiales I_0, on regroupe par un algorithme classique les composantes connexes contenues dans l'image et on leur attribue une étiquette. On stocke le résultat dans une image I'_0.

Calcul d'une image de distances : En se basant sur la distance discrète choisie, deux passes vont être réalisées sur l'image I'_0. Ces deux passes vont permettre, par application d'un masque de chanfrein, de construire deux images :
- l'image I_1 qui à chaque pixel (x,y) associe la *distance discrète* à la composante la plus proche, 0 pour les pixels appartenant à des composantes ;
- l'image I_2 qui à chaque pixel associe le *nom* de la composante la plus proche.

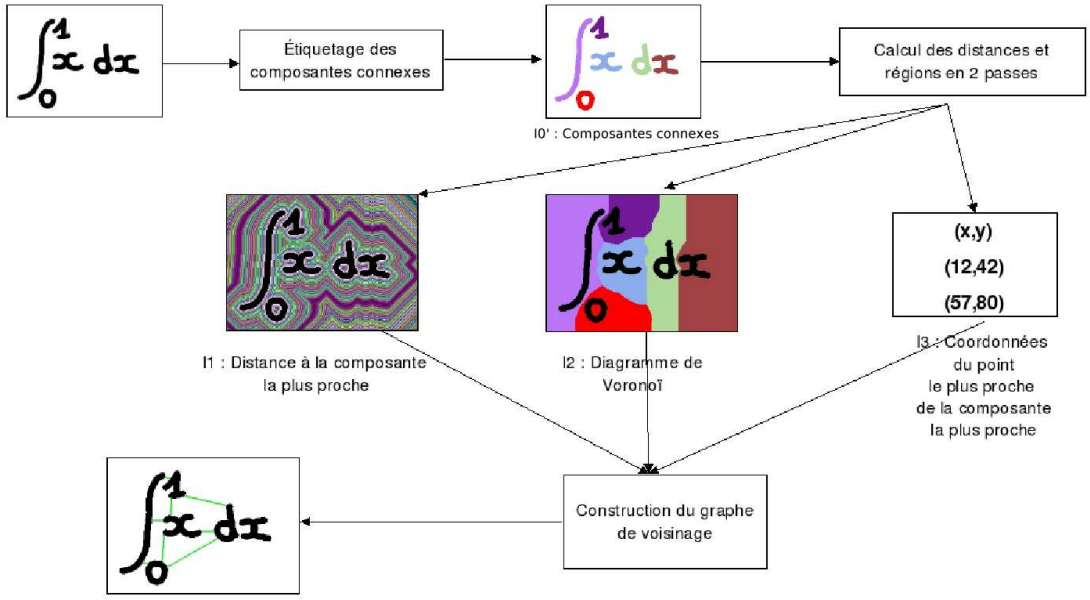


FIG. 3.4 – Schéma de la construction du graphe de voisinage [14]

On peut également mémoriser les points ayant servi à regrouper les composantes entre elles. Pour cela, on stocke dans une double matrice I_3 les coordonnées x et y des points depuis lesquels s'est propagée la distance au contour.

Extraction d'un graphe de voisinage : Les informations stockées dans I_2 donnent trivialement le pavage de Voronoï de l'image d'origine. On peut donc en déduire le graphe de Delaunay (dual du pavage de Voronoï). Les arcs de ce graphe peuvent être valués grâce aux informations contenues dans I_1, mais aussi dans les matrice I_3x et I_3y.

3.3 Choix d'implémentation

L'implémentation du graphe de voisinage offre des alternatives au niveau de la méthode de propagation des distances discrètes ainsi que de la construction du graphe.

3.3.1 Propagation des distances discrètes

Nous avons vu que la construction du graphe se base sur la propagation d'un masque de chanfrein. Cependant, les résultats obtenus peuvent être différents selon :

- la taille et les coefficients du masque de chanfrein ;
- la structure de l'image utilisée (matricielle, empans) ;
- la technique de propagation utilisée.

En effet, selon l'application que l'on fait du masque, le champ de vision entre les éléments peut être plus ou moins large.

C'est ce que nous allons voir sur l'image 3.5, composée de deux composantes connexes. Nous proposons d'étudier une méthode de propagation bidimensionnelle du masque, puis

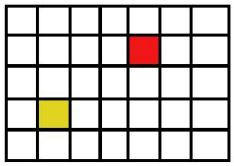


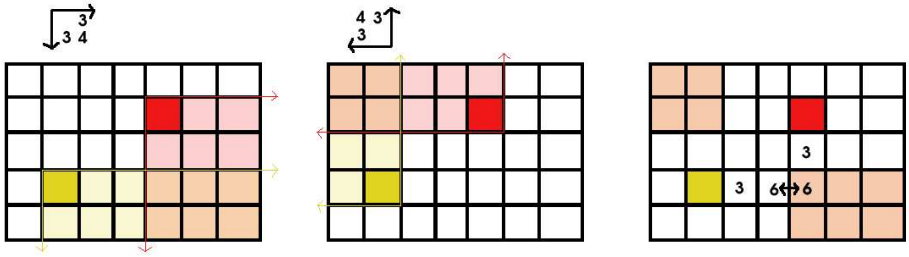
FIG. 3.5 – Image de base

une seconde technique s'appuyant sur une décomposition en deux étapes monodimensionnelles.

3.3.1.1 Propagation bidimensionnelle du masque

Pour appliquer le masque de chanfrein, on s'appuie sur une propagation des valeurs d'une ligne à l'autre, et de la gauche vers la droite. Nous allons montrer sur un exemple l'importance de la taille du masque et du cône de propagation qui en découle.

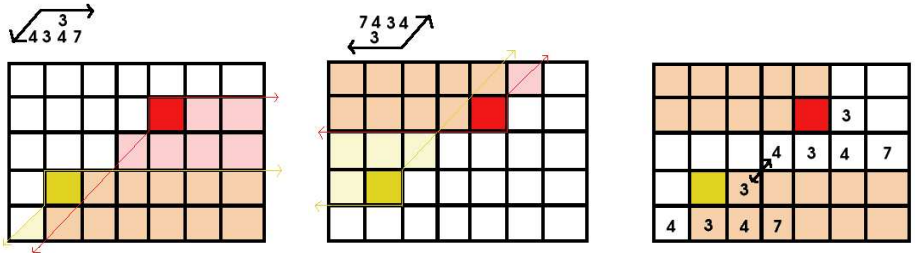
Propagation à 90 ° Appliquons un masque de chanfrein (3,4) sur la figure 3.5. Lors de la première passe (figure 3.6(a)), on réalise une propagation de haut à gauche vers le bas à droite. Le cône de propagation de chacune des composantes correspond à un angle de 90 °. Lors de la seconde passe (figure 3.6(b)), la propagation en balayage inverse fait apparaître un second cône de propagation de 90 °. Dans cet exemple précis, les deux objets ne sont pas placés dans leurs cônes de propagation. Ils ne se « voient »pas directement et connaîssent leur présence respective uniquement dans la zone orangée. Pour calculer la valeur de la distance, on la propage depuis la zone de vision commune. Avec le cône de propagation proposé, il faut suivre le chemin présenté figure 3.6(c) : au niveau de la flèche, on se trouve a une distance de 6 de chacun des deux objets ; en ajoutant 3 pour le pas horizontal, on obtient une distance totale de 15. La distance obtenue est erronée puisqu'on devrait passer d'un objet à l'autre par 3 déplacements unitaires (horizontal + diagonal*2) pour un coût de 11.



(a) Première passe (b) Seconde passe (c) Calcul des distances

FIG. 3.6 – Application d'un masque de chanfrein (3,4)

Propagation à 135 ° On peut corriger ce défaut en élargissant le champ d'application du masque. En effet, plus on applique un masque de chanfrein large, plus on augmente le champ de vision. Ainsi, en appliquant un masque (3,4,7), le cône de vision passe à 135 °, comme montré sur les figures 3.7(a) et 3.7(b). Lors du calcul de la distance, figure 3.7(c), on se base au niveau de la flèche. On se trouve respectivement à une distance de 3 et 4 de chacun des objets. A ces valeurs, on ajoute le coût d'un déplacement en diagonale pour obtenir une distance totale de 11, ce qui correspond bien à la valeur théorique.



(a) Première passe (b) Seconde passe (c) Calcul des distances

FIG. 3.7 – Application d'un masque de chanfrein (3,4,7)

Elargissement du cône de propagation On peut, de manière générale, choisir le cône de propagation souhaité, découlant de la taille du masque choisi. Cependant, plus le masque de chanfrein et le cône de propagation sont larges, plus les valeurs sont précises mais plus cela implique un coût de calcul important.

3.3.1.2 Décomposition de la propagation

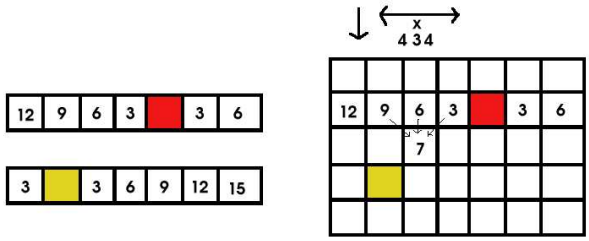
Une seconde méthode de propagation consiste à appliquer dans un premier temps le masque sur les lignes de l'image, puis à fusionner verticalement ces informations. Ceci est particulièrement adapté à une structure particulière d'image sous forme d'empans. Une image d'empans est décomposée en un ensemble de lignes. Pour chaque ligne, on connaît la succession des pixels noirs de cette ligne. On peut donc calculer ligne par ligne, pour les pixels blancs, la distance au pixel noir le plus proche, en appliquant le coefficient horizontal du masque de chanfrein. L'exemple de la figure 3.8(a) présente ce calcul des coefficients par ligne.

L'étape de propagation consiste alors à fusionner les lignes horizontales en regardant pour chaque pixel les valeurs contenues dans les trois pixels supérieurs ou inférieurs selon le sens de propagation. Sur la figure 3.8(b), on trouve la valeur 7 donnée dans l'exemple par propagation de la ligne supérieure : il s'agit en fait de la valeur minimale entre (9+4, 6+3, 3+4).

Pour cette méthode de propagation, les distances obtenues correspondent à celles attendues.

3.3.1.3 Implémentation retenue

Ces quelques exemples montrent que, du choix de la structure d'image et du masque de chanfrein, dépendent les valeurs trouvées.



(a) Propagation linéaire (b) Fusion des lignes

FIG. 3.8 – Propagation d'un masque (3,4) sur une image d'empans

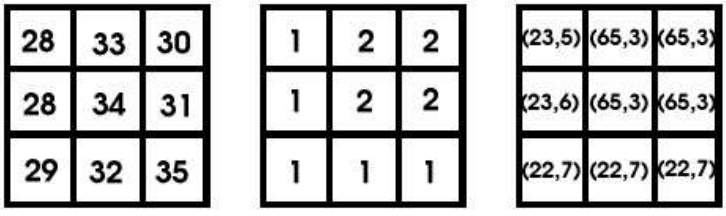
Dans notre cas, nous avons choisi l'implémentation basique du masque de chanfrein (3,4) sur une image matricielle, telle que présentée paragraphe 3.3.1.1, avec un cône de 90 °. En effet, dans le déroulement de l'algorithme, on souhaite stocker les images I_1, I_2, I_3, qui sont des images de valeurs et pour lesquelles les images matricielles sont plus adaptées que des images d'empans. D'autres part, l'information principale qui nous intéresse dans le graphe est l'existence où non d'un lien entre deux composantes, et quels en sont les points d'ancrage. La valeur finale de la distance est moins importante dans la mesure où, dans l'utilisation que nous faisons du graphe, nous pouvons recalculer une distance euclidienne à partir des extrémités du lien.

3.3.2 Extraction du graphe

Lorsque les images I_1, I_2, I_3 ont été extraites depuis l'image d'origine, l'étape suivante de l'algorithme consiste à construire le graphe de voisinage. La méthode consiste à parcourir l'image I_2 contenant, pour chaque pixel, le nom de la composante la plus proche. A chaque fois que l'on se trouve sur un pixel à la frontière entre deux zones d'influence de composantes, on ajoute un lien dans le graphe.

Nous allons voir un exemple plus précis.

3.3.2.1 Exemple de construction



(a) I_1 : distance à la composante la plus proche

(b) I_2 : nom de la composante la plus proche

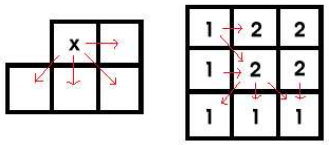
(c) I_3 : coordonnées du point le plus proche

FIG. 3.9 – Exemples d'images extraites

On suppose que la construction des images I_1, I_2, I_3, a donné localement le résultat présenté figure 3.9. L'image est décomposée en deux zones d'influence, celle de la compo-

sante 1 et celle de la composante 2 qui sont voisines. Le but est donc d'extraire les arcs du graphe reliant ces deux composantes.

Pour extraire le graphe, on va parcourir l'image I_2 et regarder pour chaque pixel si celui-ci se trouve a une frontière dans une des directions décrites figure 3.10(a). A chaque fois que le pixel se trouve à une frontière, on va ajouter le lien correspondant dans le graphe. Comme le montre la figure 3.10(b), on peut détecter ici 7 directions dans lesquelles il y a une frontière entre les zones 1 et 2. On va créer autant d'arcs, étiquetés grâce aux images I_1 et I_3, recensés ici dans le tableau 3.1.



(a) (b)

FIG. 3.10 – Parcours de l'image I_2 pour l'extraction du graphe.

Composantes A et B	Point d'ancrage sur A	Point d'ancrage sur B	Distance
1 - 2	(23,5)	(65,3)	28+33+3 = 64
1 - 2	(23,5)	(65,3)	28+34+4 = 66
1 - 2	(23,6)	(65,3)	28+34+3 = 65
1 - 2	(22,7)	(65,3)	29+34+4 = 67
1 - 2	(22,7)	(65,3)	32+34+3 = 69
1 - 2	(22,7)	(65,3)	35+34+4 = 73
1 - 2	(22,7)	(65,3)	35+31+3 = 69

TAB. 3.1 – Lien contenus dans le graphe de voisinage

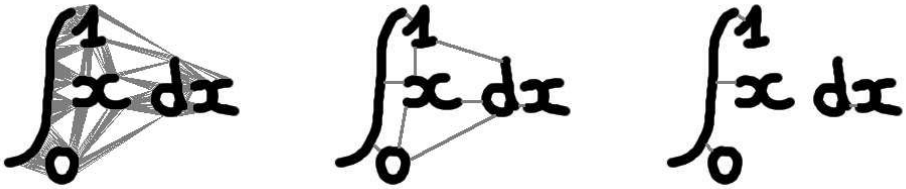
Cet exemple montre que pour une simple zone locale de 9 pixels, on obtient déjà 7 arcs dans le graphe, entre les composantes 1 et 2, avec des distances variables selon les points d'ancrage des liens. On appelle par la suite *graphe complet* ce graphe contenant tous les liens extraits directement des images.

3.3.2.2 Implémentation retenue

Nous avons vu que le graphe complet initialement construit contient un très grand nombre d'arcs. Ainsi, sur l'exemple de la figure 3.11(a), on compte 2680 arcs pour seulement 6 composantes connexes. On peut donc vouloir travailler sur un graphe élagué selon une des deux méthodes suivantes :

- entre deux composantes, on ne conserve que le lien ayant la distance la plus courte (figure 3.11(b)) ;
- pour chaque composante, on ne conserve que le lien vers la composante la plus proche (figure 3.11(c)).

La principale information qui nous intéresse dans le graphe est l'existence ou non d'un lien entre deux composantes, ainsi que les coordonnées de l'arc de plus courte distance. Nous choisissons donc de travailler sur le graphe réduit présenté figure 3.11(b).



(a) Graphe complet : 2680 arcs

(b) Graphe de plus courte distance : 9 arcs

(c) Graphe au plus proche voisin : 6 arcs orientés

FIG. 3.11 – Variantes de graphes

Toutefois, les informations contenues dans le graphe réduit étant restreintes, nous insistons sur l'intérêt de garder à disposition les images I_1, I_2 et I_3 pour pouvoir, au besoin, extraire localement d'autres informations.

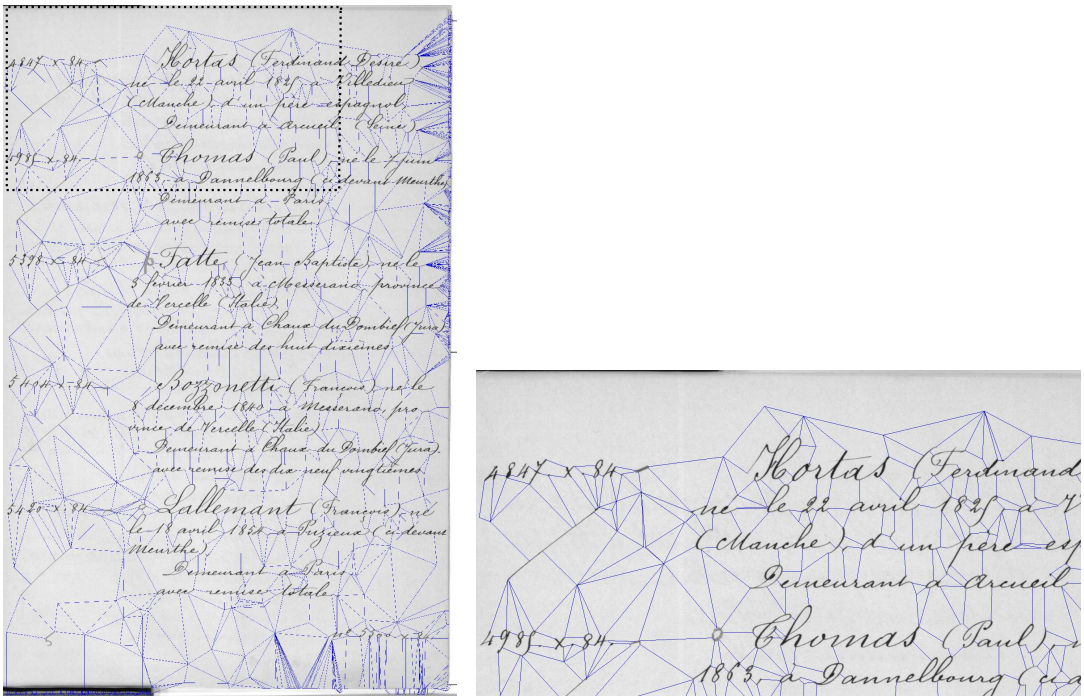
3.4 Résultats obtenus

Nous présentons ici un exemple de résultat obtenu sur une des images utilisées par notre exemple d'application (chapitre 5). L'image 3.12 et le tableau associé 3.2 présentent les statistiques de création du graphe. Les temps de calculs correspondent à ceux d'un processeur Pentium 4 cadencé à 2600MHz, avec une mémoire primaire de 1024MB.

```
Statistiques sur le graphe et l'image
=====================================================
Calcul des images I1, I2, I3x, I3y
Temps de calcul :                                    2.3s
-----------------------------------------------------
Taille de l'image :                                  2032 * 3095
Nombre de composantes connexes :                     1227
Nombre de pixels appartenant a une composante connexe :  249060
Nombre moyen de pixels par composantes connexes :    202.983
Pourcentage de pixels faisant partie d'une CC :      3.96022%
Nombre de pixels formant les frontieres des CC :     178068
Pourcentage de pixel de l'image formant les frontieres de CC :  2.8314%
-----------------------------------------------------
Creation du graphe reduit
Temps de calcul :                                    0.33s
-----------------------------------------------------
Le graphe est le graphe reduit
Nombre de liens du graphe :                          3427
Nombre moyen de liens par composantes connexes :     2.79299
Nombre de pixels distincts utilises pour le graphe :  4302
Pourcentage de pixels des frontieres utilises pour le graphe :  2.41593%
Nombre moyen de liens partant d'un pixel utilisé :   1.59321
-----------------------------------------------------
Temps de calcul du graphe complet :                  3s
-----------------------------------------------------
Statistiques sur le graphe complet
Nombre de liens du graphe :                          432685
Nombre moyen de liens par composantes connexes :     352.637
Nombre de pixels distincts utilises pour le graphe :  33412
Pourcentage de pixels des frontieres utilises pour le graphe :  18.7636%
Nombre moyen de liens partant d'un pixel utilisé :   25.9
=====================================================
```

TAB. 3.2 – Exemple de statistiques de construction du graphe

Cet exemple est représentatif des résultats obtenus. Le calcul des images a une complexité en $O(n_P)$ où n_P est le nombre de pixels de l'image. D'après les résultats obtenus, le calcul du graphe réduit a une complexité moyenne de l'ordre de $O(n_P n_C)$ où n_C est le nombre de composantes connexes. Le graphe réduit nécessite un temps de calcul beaucoup plus faible que le graphe complet ; c'est aussi un de ses intérêts.



(a) (b) Zoom

FIG. 3.12 – Exemple d'image traitée par un graphe réduit selon la méthode de la figure 3.11(b)

Chaque composante connexe est en moyenne liée à deux ou trois autres composantes, ce qui correspond au fait qu'on étudie des lignes de texte et que chaque composante est liée avec les lettres voisines.

On constate également que les pixels qui servent d'ancrage à un lien sont en moyenne utilisés plus d'une fois. Ce résultat est encore plus flagrant sur le graphe complet puisqu'en moyenne, seulement un pixel sur cinq est utilisé par le graphe, et il sert dans ce cas à 25.9 liens. Ces pixels proéminents pourraient donc être considérés comme caractéristiques de la composante.

CHAPITRE

4

Réalisation de l'intégration des graphes de voisinage dans la méthode DMOS

Après la mise en oeuvre d'un graphe de voisinage, la seconde partie du stage a consisté à intégrer ce graphe dans la méthode DMOS. Le but de ce travail est de pouvoir, lors de la phase de détection de terminaux dans l'analyse d'un document, s'appuyer sur le graphe de voisinage.

Comme nous avons vu dans la partie 1.3.4, l'analyse des terminaux passe par l'utilisation conjointe d'un opérateur de position tel que `AT`, et d'un détecteur de terminaux tel que `TERM_CMP`. Ces opérateurs s'appuient sur les notions de point d'ancrage, de zone de recherche et de distances avec des boîtes englobantes.

Le travail d'intégration du graphe consiste donc à ajouter de nouveaux opérateurs, afin de remplacer le mécanisme de distance entre point d'ancrage et boîte englobante par une recherche de voisinage dans le graphe. Pour définir ces opérateurs, il faut être capable de préciser comment ils interviennent par rapport au point d'ancrage et à la zone de recherche. Le graphe de voisinage doit bien sûr être accessible tout au long de l'analyse.

Dans un premier temps, nous nous sommes donc intéressés à la manière dont on pouvait manipuler le graphe au cours de l'analyse. Puis, après avoir choisi les mécanismes qui s'appliqueraient pour la détection des terminaux, nous avons mis en place les éléments nécessaires dans DMOS.

4.1 Construction et manipulation du graphe

Nous présentons quelles sont les contraintes liées à l'introduction du graphe dans DMOS avant de présenter la solution retenue.

4.1.1 Contraintes de manipulation

Le graphe de voisinage est implémenté en C++ alors que le noyau de DMOS est codé en λProlog. Il faut donc manipuler, depuis DMOS, une référence vers le graphe.

La structure de graphe de voisinage n'est pas aussi complète que les images I_1, I_2, I_3, et certaines informations contenues dans ces images peuvent être utiles. C'est pourquoi, on souhaite non seulement manipuler le graphe de voisinage mais garder également l'accès aux images construites.

Enfin, on envisage la possibilité de regrouper les éléments contenus dans le graphe au cours de l'analyse. Il faut donc être capable de gérer une hiérarchie de graphes et de récupérer une version précédente du graphe en cas de retour arrière dans l'analyse Prolog.

4.1.2 Solution réalisée

Nous avons mis en oeuvre une solution respectant les contraintes ci-dessus.

4.1.2.1 Manipulation

Dans la version précédente, DMOS manipulait tout au long de l'analyse un lien vers un objet C++ de type `DocStru`. Cet objet permet, entre autres, de stocker les pixels de l'image traitée, d'extraire les composantes connexes et les segments de l'image. En outre, cet objet est prévu pour manipuler les images de manière hiérarchique : on peut ainsi envisager de changer la résolution ou la zone de l'image traitée au cours de l'analyse.

Nous avons donc choisi de nous baser sur cet objet manipulé pour intégrer le graphe de voisinage. Nous avons introduit un nouveau type d'objet en C++, `ImageVoronoi`, héritant de `DocStru`, et contenant à la fois les images I_1, I_2, I_3, et le graphe réduit. C'est cette structure qui est maintenant manipulée depuis λProlog. Si on souhaite gérer une hiérarchie de graphe, c'est possible au même titre que l'on pouvait précédemment gérer une hiérarchie d'images.

4.1.2.2 Construction

L'appel de la construction du graphe de voisinage est réalisé depuis λProlog lors de l'initialisation de l'analyse, après que l'étiquetage des composantes connexes ait été réalisé. On offre la possibilité à l'analyseur d'écarter des composantes qui sont jugées comme du bruit. L'appel de la construction du graphe peut donc se faire soit sur l'image initiale, soit en restreignant les composantes manipulées.

4.2 Mécanisme de détection

Disposant du graphe de voisinage tout au long de l'analyse, il faut maintenant déterminer comment celui-ci va être utilisé pour la détection des terminaux.

4.2.1 Structures internes manipulées

Le principe de détection classique des terminaux s'appuie sur une zone de recherche et un point d'ancrage. Il nous faut déterminer ce qu'il advient de ces éléments dans la reconnaissance basée sur le graphe.

4.2.1.1 Principe de la détection classique

Dans l'utilisation habituelle de DMOS, la recherche de terminaux se fait par la combinaison d'un opérateur de position `AT` ou `AT_ABS` et d'un détecteur de terminaux `TERM_SEG` ou `TERM_COMP`. Ces opérateurs utilisent une structure appelée `curseur`, qui est composé de :

- les coordonnées du point d'ancrage ;
- le prochain élément analysé ;

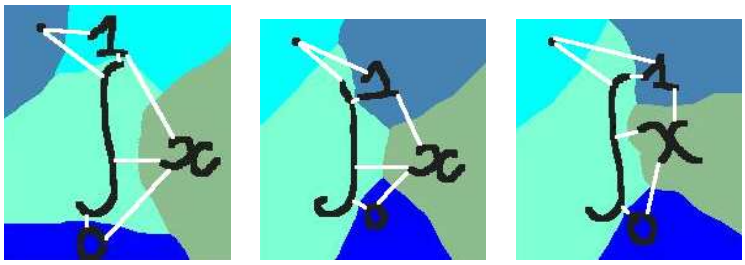
– une zone de recherche.

L'utilisation des opérateurs de position permet de modifier le point d'ancrage et la zone de recherche. À l'appel d'un détecteur de terminaux, on liste les composantes présentes dans la zone, puis on recherche celle qui est la plus proche du point d'ancrage.

4.2.1.2 Détection grâce au graphe

Nous étudions ici l'utilité de conserver les notions de zone et de point d'ancrage dans la recherche basée sur le graphe.

Zone de recherche La zone de recherche est fixée par les opérateurs de position. Si on la supprime, il faut être capable de préciser, grâce au graphe, le positionnement de la recherche. Nous allons étudier quelques solutions en s'appuyant sur la figure 4.1.



(a) (b) (c)

FIG. 4.1 – Exemples d'orientations variables des liens pour un même type d'image

On s'intéresse à la détection du 1 en partant du signe intégrale, on voudrait donc chercher « en haut à droite » de l'intégrale. Avec un hypothétique opérateur de position lié au graphe, `AT_GRAPH`, la règle EPF correspondante s'exprimerait :

```
signeIntegrale &&
AT_GRAPH(hautDroite signeIntegrale) &&
borneSup
```

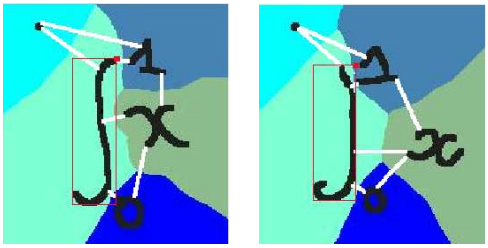
Le problème consiste à exprimer la signification de `AT_GRAPH`. Cet opérateur pourrait s'appuyer sur l'orientation des liens partant de `signeIntegrale`, ou bien de la position des points de départs de ces liens sur `signeIntegrale`.

Si on regarde la direction du lien, sur la figure 4.1(b), l'arc entre l'intégrale et le 1 a bien une direction vers en haut à droite. Par contre, sur la figure 4.1(a), ce lien est dirigé légèrement vers la gauche, et sur la figure 4.1(c), il n'est pas orienté vers le haut. En outre, sur la figure 4.1(c), il existe un lien partant de l'intégrale vers le haut droite, mais qui relie le signe x. Ces contre-exemples montrent que, dans un cas général, on ne peut pas se baser sur l'orientation du lien pour rechercher un élément à partir d'un autre.

La position des points de départ des liens sur l'intégrale varie également d'une image à l'autre. Partant du signe intégrale vers le x, le lien peut être ancré dans la partie supérieure de l'intégrale (figure 4.1(c)) comme dans la partie inférieure (figure 4.1(b)). Là encore, on ne peut pas, dans un cas général, s'appuyer uniquement sur la position du début du lien pour fixer une direction de recherche.

Compte tenu de ces observations, une solution efficace pour analyser le document dans une direction donnée reste la fixation d'une zone de recherche par un opérateur de position. Nous conservons donc ce mécanisme.

Point d'ancrage Dans la méthode d'analyse classique, le point d'ancrage est fixé par l'opérateur de position, de manière arbitraire, connaissant les coordonnées de la composante précédemment reconnue. Par exemple, dans la figure 4.3, le point d'ancrage pour la recherche du 1 est l'extrémité en haut à droite de la boîte englobante de l'intégrale.



(a) (b)

FIG. 4.2 – Mise en place du point d'ancrage

Pour appliquer la recherche de composante avec le graphe, on souhaiterait non plus se baser sur un point d'ancrage mais sur une composante. En effet, on veut chercher le plus proche voisin d'une composante et non d'un point.

La première solution consiste à conserver la notion de point d'ancrage. Lorsqu'on effectue une recherche, on récupère le nom de la composante qui a généré la zone d'influence dans laquelle se situe le point. On effectue la recherche à partir de cette composante. Dans le cas de la figure 4.2(a), le point d'ancrage appartient bien à la zone d'influence de l'intégrale. C'est à partir de cette composante qu'on va continuer l'analyse. Par contre, dans le cas de la figure 4.2(b), le point d'ancrage, déterminé par la boîte englobante de l'intégrale, appartient à la zone d'influence du 1. Dans ce cas, l'analyse va continuer en se basant sur le 1, ce qui ne correspond pas au mécanisme attendu. Cette solution ne convient donc pas dans le cas général.

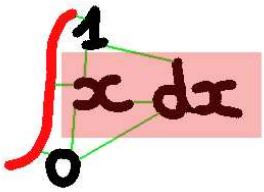
Nous avons donc choisi d'offrir la possibilité de mémoriser le nom d'une composante de référence. Nous modifions la structure du `curseur`, et plus précisément la partie `prochain élément analysé`, en introduisant un `élément de référence`. Cette composante de base peut être spécifiée par l'utilisateur grâce à l'opérateur de position.

Si l'utilisateur ne précise pas cette composante de base, alors on souhaite appliquer la première méthode pour retrouver une composante à partir du point d'ancrage. C'est pourquoi, en plus de modifier la structure du curseur, nous conservons la notion de point d'ancrage.

4.2.2 Tri des éléments dans une zone

Disposant des concepts d'élément de référence, de point d'ancrage et de zone de recherche, il faut maintenant fixer les règles du calcul de la distance, pour le tri des composantes de la zone par ordre de proximité. On remarque que les composantes contenues dans la zone ne sont pas nécessairement liées, dans le graphe, à la composante de référence.

Par exemple, sur la figure 4.3(a), on cherche à trier les composantes situées dans la zone, x et dx par rapport au signe *intégrale*.



(a)

FIG. 4.3 – Exemple de tri par distance des composantes dans une zone

On décompose le problème de calcul des distances selon que l'on dispose de :

- une composante de référence reliée par un arc aux composantes à trier (composante x) ;
- une composante de référence non reliée par un arc aux composantes à trier (composante dx) ;
- un point d'ancrage (si la composante de référence *intégrale* n'était pas spécifiée).

Pour la composante x, on dispose directement dans le graphe de voisinage de la distance discrète entre deux composantes voisines. On pourrait donc utiliser ces distances pour trier les composantes. Dans le second cas, en revanche, il faut calculer un chemin entre la composante de référence et les éléments de la zone. On ne dispose plus alors de distance discrète pour ce chemin.

Nous avons donc choisi d'effectuer le tri des composantes dans la zone en se basant sur une distance euclidienne recalculée à partir des extrémités des chemins. Pour chacun des cas présentés précédemment, nous calculons donc la distance euclidienne respectivement entre :

- les extrémités de l'arc reliant la composante de base et l'élément à examiner ;
- les extrémités du chemin reliant la composante de base et l'élément à examiner ;
- le point d'ancrage et l'extrémité du chemin entre la composante d'influence du point d'ancrage et l'élément à examiner.

Ainsi, sur l'exemple, la distance entre l'*intégrale* et la composante x est calculée à partir des extrémités de l'arête du graphe reliant ces deux éléments. La distance entre l'*intégrale* et la composante dx est extraite grâce aux points extrémités du chemin entre ces deux composantes.

4.3 Eléments mis en place

Une fois le mécanisme de reconnaissance défini, nous avons mis en place les nouveaux éléments nécessaires à l'utilisateur pour pouvoir écrire une grammaire en utilisant le graphe de voisinage. Il s'agit principalement d'un nouvel opérateur de reconnaissance de terminaux `TERM_CMP_GRAPH`, mais aussi d'outils liés au traitement du graphe.

Nous avons réalisé les tests unitaires de ces éléments sur la grammaire décrivant les intégrales qui avait servi à prendre en main DMOS, présentée au paragraphe 1.3.2.

4.3.1 Opérateur *TERM_CMP_GRAPH*

Nous présentons le fonctionnement global de l'opérateur avant de détailler un point précis de son implémentation.

4.3.1.1 Fonctionnement global

L'implémentation de l'opérateur `TERM_CMP_GRAPH` est globalement basée sur celle de `TERM_CMP` (opérateur présenté paragraphe 1.3.3.3). Sa syntaxe est quasiment identique, au détail près que l'on peut appliquer plusieurs pré conditions lors de la recherche. Cela peut permettre, par exemple, de combiner des conditions liées au graphes, telles que celles présentées paragraphe 4.3.4, avec des conditions existant auparavant. La syntaxe de l'opérateur est donc la suivante :

```
TERM_CMP_GRAPH ListeDePreConditions PostCondition Etiquette ComposanteReconnue
```

L'effet de cet opérateur est de rechercher un élément :
- dans la zone de recherche ;
- vérifiant chacune des pré conditions de la liste `ListeDePreConditions` ;
- le plus proche de la composante de référence d'après la distance extraite du graphe (tri détaillé paragraphe 4.2.2).

La composante trouvée `ComposanteReconnue` doit alors vérifier la postcondition `PostCondition`, sinon l'opérateur échoue.

4.3.1.2 Recherche des éléments dans une zone

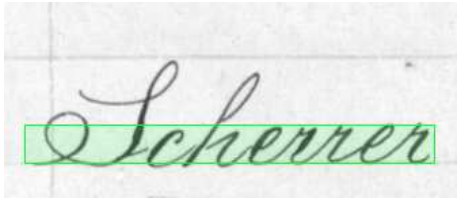
La première action réalisée par `TERM_CMP_GRAPH` consiste à lister les éléments inclus dans la zone de recherche. Cette étape donne lieu à un choix d'implémentation, et nous avons testé deux versions.

Recherche basée sur les boîtes englobantes Dans un premier temps, nous avons utilisé la méthode de recherche de `TERM_CMP`, qui consiste à parcourir toutes les composantes à analyser, puis à tester si un des points caractéristiques de cette composante appartenait à la zone. Ces points caractéristiques sont ici les milieux des côtés des segments de la boîte englobante de la composante.

Ainsi, sur la figure 4.4, cherchons les composantes comprises dans la zone de recherche dessinée figure 4.4(a). Ici, 4 composantes nous intéressent : le *s*, le *c*, le début du *h* et le *errer*.

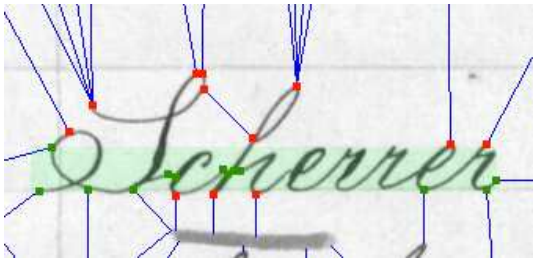
Calculons alors, pour chacune des composantes, les points des milieux des côtés des boîtes englobantes (figure 4.4(b)). Les points inclus dans la zone de recherche apparaissent en vert, les autres en rouges. Pour chaque point inclus dans la zone, on considère que la composante associée appartient à la zone. Avec cette méthode de détection, on n'obtient que 2 composantes dans la zone : le *c*, et le *errer*.

Recherche basée sur le graphe Nous avons vu dans le paragraphe 3.4 que les points des composantes qui servaient de départ aux arêtes du graphe étaient intéressant. Nous mettons donc en place une autre méthode de détection des éléments qui se base sur la présence ou non de ces points dans la zone.



(a) Zone de recherche

(b) Recherche avec les boîtes



(c) Recherche avec le graphe

FIG. 4.4 – Recherche de composantes dans une zone

Ainsi, dans la figure 4.4(c), chacune des 4 composantes recherchées a au moins une extrémité de lien dans la zone (symbolisée par un point vert). Elles sont donc toutes reconnues comme faisant partie de cette zone.

Implémentation finale Dans la version finale de notre implémentation, une composante appartient à une zone si :

- un point caractéristique du graphe appartient à la zone ;
- et sinon si :
  - un point caractéristique de la boîte englobante appartient à la zone.

4.3.2 Opérateurs de position

Nous avons vu dans le paragraphe 4.2.1.2 que l'on souhaitait mémoriser la composante servant de référence pour la recherche de l'élément suivant. Ceci se passe au niveau des opérateurs de position. Il n'y a pas de grand changement à ce niveau puisqu'au lieu de modifier uniquement la zone de recherche et le point d'ancrage, l'opérateur doit affecter en outre le prochain élément à analyser.

Du point de vue de l'utilisateur d'EPF, la syntaxe reste donc la même que celle décrite au paragraphe 1.3.3.1.

Au niveau de l'implémentation, le travail a consisté à rendre possible l'encapsulation d'une composante de référence avec le prochain élément à analyser.

4.3.3 Opérateur *FIND_GRAPH UNTIL*

L'opérateur `FIND`, décrit au paragraphe 1.3.3.5, a pour but d'essayer d'appliquer une règle sur tous les éléments présents dans la zone, en les testant par ordre de distance. Il semble donc nécessaire de définir la version analogue qui va tester les éléments en les prenant dans l'ordre des distances créé par le graphe. L'opérateur `FIND_GRAPH` a donc une syntaxe semblable à celle de `FIND` :

```
FIND_GRAPH(Règle) UNTIL(ConditionArret)
```

On essaye d'appliquer la `Règle` jusqu'à succès ou la condition `ConditionArret` (échec).

4.3.4 Nouvelles conditions

Grâce aux informations contenues dans le graphe, nous mettons en place de nouvelles conditions qui pourront être utilisées comme pré ou post conditions des détecteurs de terminaux.

4.3.4.1 Existence de lien direct

La condition `condExisteLienDirect` a la syntaxe suivante :

```
condExisteLienDirect compReference compAAnalyser
```

Elle permet de vérifier que la composante à analyser `compAAnalyser` est liée par une arête, dans le graphe de voisinage, à la composante de référence `compReference`. Cette condition est utile pour rechercher des composantes qui soient nécessairement voisines les unes des autres.

Elle permet également de gérer la notion de masquage de composantes par une autre, dans la reconnaissance de colonnes par exemple. Ainsi, sur l'extrait de journal présenté figure 4.5, on cherche à détecter les lignes successives dans la colonne de gauche. On définit une ligne comme une succession de lettres liées par un lien direct, récursivement depuis la première lettre, selon la grammaire suivante :

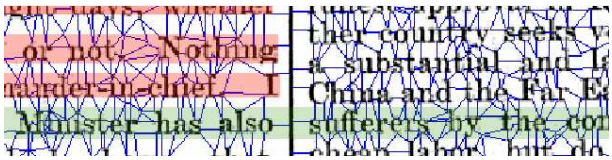
```
ligne ::=
    premiereLettre &&
    AT(droite premiereLettre) &&
    suiteDeLettresLiees premiereLettre.

suiteDeLettresLiees premiereLettre ::=
    TERM_CMP_GRAPH (condExisteLienDirect premiereLettre) LettreSuivante &&
    AT(droite LettreSuivante) &&
    suiteDeLettresLiees LettreSuivante.
```

Dans cette grammaire, `premiereLettre` est un terminal. Nous avons volontairement omis certains paramètres de `TERM_COMP_GRAPH` ainsi que la règle d'arrêt de la récurrence, afin de simplifier la lecture.

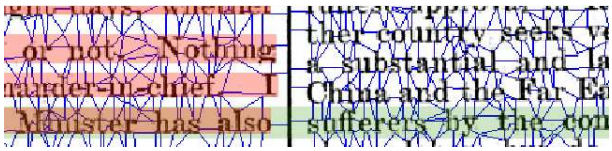
Cette définition permet de détecter la fin de la ligne lorsqu'on « bute » sur une colonne et que la lettre suivante n'est pas liée par un lien direct, c'est à dire lorsque l'élément suivant lié par le graphe est un segment et non une lettre.

4.3.4.2 Orientation du chemin

Nous avons vu dans le paragraphe 4.2.1.2 que l'orientation des arcs du graphe ne suffisait pas à déterminer une zone de recherche dans le cas général. En revanche, dans un cas particulier, il peut être utile de connaître le positionnement relatif de la composante cherchée par rapport à une composante de référence. Nous mettons donc en place les conditions suivantes :



(a) Zone de recherche de la ligne suivante



(b) Fin de ligne détectée par le trait de colonne

FIG. 4.5 – Recherche des lignes dans la colonne de gauche de l'extrait de journal

```
condADroiteGraphe   compReference   compAAnalyser
condAGaucheGraphe   compReference   compAAnalyser
condAuDessousGraphe compReference   compAAnalyser
condAuDessusGraphe  compReference   compAAnalyser
```

Pour vérifier ces conditions, on calcule le chemin entre la composante de référence `compReference` et la composante à analyser `compAAnalyser`. La condition réussit si l'extrémité du chemin est située respectivement à droite, à gauche, au dessous ou au dessus de l'origine. Il s'agit bien ici de la recherche d'un chemin et non d'une arête ; cela permet d'éviter la redondance avec la condition précédente.

4.3.4.3 Distance maximale

La condition `condDistanceMax` a la syntaxe suivante :

```
condDistanceMax compReference distanceMax compAAnalyser
```

Elle permet de vérifier que la composante à analyser `compAAnalyser` n'est pas située à une distance plus grande que `distanceMax` de la composante de référence `compReference`. Cette condition permet de détecter des composantes qui soient suffisamment proches.

En réalité, elle existe selon deux variantes :

- `condDistanceLienMax` permet d'étudier la distance discrète correspondant à la valeur de l'arc entre les deux composantes, et échoue si les composantes ne sont pas liées par un arc dans le graphe ;
- `condDistanceEuclidMax` calcule la distance euclidienne entre les composantes, qu'elles soient liées dans le graphe ou non.

4.3.4.4 Composante la plus liée

On dispose d'une information liée au graphe complet : le taux de liaison entre deux composantes. En effet, on peut calculer, pour une composante de référence, son taux de liaison avec chacune des composantes. Ce taux correspond au pourcentage de frontière de zone de Voronoï commune.

Par exemple, sur la figure 4.6, on peut calculer le taux de liaison de *1* (composante 1), *x* (composante 4) et *0* (composante 6) par rapport à l'*intégrale* (composante 2). Les résultats sont présentés dans le tableau 4.1

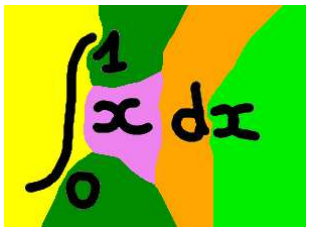


FIG. 4.6 – Frontières communes et taux de liaison

```
-----------------------------------------------
Taux de liaison de la composante 2 vers 1 :     35.2406%
-----------------------------------------------
Taux de liaison de la composante 2 vers 4 :     27.9584%
-----------------------------------------------
Taux de liaison de la composante 2 vers 6 :     36.801%
-----------------------------------------------
```

TAB. 4.1 – Exemple de calcul de taux de liaison

Nous proposons donc la condition `condCompPlusLiee` qui permet de trouver, dans une zone, la composante la plus liée à une composante de référence. Sa syntaxe est la suivante :

```
condCompPlusLiee   compReference   compPlusLiee
```

Cette condition réussit si la composante `compPlusLiee` est la composante de la zone qui a la plus fort taux de liaison avec la composante de référence `compReference`.

4.3.5 Informations statistiques

L'utilisation des graphes de voisinage nous a permis de mettre en place un extracteur de statistiques sur des distances. En effet, nous avons vu dans le paragraphe 2.3 que les principales utilisations du graphe de Voronoï se basaient sur l'apprentissage de seuils de distances. Cet apprentissage s'effectue de manière statistique sur l'ensemble de la page de document, ce qui restreint les applications à des documents relativement réguliers quant aux distances inter-lignes et inter-mots.

Grâce à la méthode DMOS, nous avons la possibilité de spécifier un contexte de recherche et donc de restreindre l'apprentissage statistique des seuils à une zone locale. Nous proposons donc un outil capable d'apprendre un seuil de distances dans une zone spécifique de l'image, et selon une direction privilégiée des liens.

L'utilisateur précise la zone de recherche sur laquelle il souhaite établir ses statistiques. Les distances des arêtes du graphe, contenues dans cette zone, sont alors extraites.

L'utilisateur peut restreindre la liste des arêtes utilisées pour le calcul statistique selon un des critères suivants :

- `*` tous les arcs sont sélectionnés ;
- `-` on n'utilise que les arêtes orientées horizontalement. Ceci peut être utile pour la détection de mots dans la mesure ou les lettres sont alignées de gauche à droite ;
- `|` on n'utilise que les arêtes orientées verticalement ;

– `2` on utilise, pour chaque composante, uniquement le lien avec ses deux plus proche voisin. Ceci est également utilisé pour la détection des mots puisque les lettres sont liées prioritairement à leurs voisins de droite et de gauche.

Les distances mémorisées sont, au choix de l'utilisateur, soit la distance discrète étiquetant l'arc du graphe entre les deux composantes, soit une distance euclidienne recalculée.

Disposant d'une liste de distances, on peut alors extraire les statistiques suivantes :
– la moyenne ;
– la médiane ;
– le seuil de séparation des classes par apprentissage non supervisé avec la méthode des k-moyennes (k = 2) ;
– le seuil de séparation des classes par apprentissage non supervisé avec la méthode des k-médianes (k = 2).

Ces résultats peuvent être utilisés avec la condition `condDistanceMax` afin de détecter des composantes espacées d'une distance inférieure au seuil calculé.

4.4 Perspectives

4.4.1 Positionnement de non-terminaux

Actuellement, dans l'utilisation du graphe, on utilise la notion de voisinage uniquement entre deux composantes connexes. On aimerait pouvoir positionner, de la même manière, des non-terminaux. Par exemple, on pourrait spécifier le fait qu'un paragraphe soit composé de lignes les unes en dessous des autres. Pour cela, de la même manière que l'on spécifie une composante de référence dans le curseur pour effectuer la recherche du terminal suivant, on pourrait mémoriser un non-terminal en tant qu'objet de référence. Si on connaît l'ensembles des éléments constituant ce non-terminal, on peut, par une opération de regroupement des composantes dans le graphe, récupérer les informations habituelles de voisinage.

4.4.2 Hiérarchie de graphes

La notion de hiérarchie de graphes est liée à celle de la hiérarchie d'images. Comme nous avons vu dans le paragraphe 4.1.2.1, on peut envisager de faire évoluer la résolution et la zone de l'image utilisée en cours d'analyse. Dans ce cas, il faut que le graphe associé évolue également, afin de s'adapter à la structure analysée.

4.4.3 Informations statistiques

Il pourrait être utile de mettre en place d'autres outils de traitement local du graphe. On pourrait par exemple s'appuyer sur une construction du sous-graphe local à la zone et travailler sur des techniques de recherche d'arbres couvrant de poids minimal, et autres analyses de graphes.

CHAPITRE

5

Exemples d'application

Nous avons validé les outils de DMOS liés au graphe de voisinage selon plusieurs axes. Nous avons tout d'abord appliqué les opérateurs à l'exemple simple de la grammaire des intégrales présentée dans le premier chapitre. Nous avons ensuite profité du fait que l'analyse de DMOS se réalise selon le contexte afin de mettre en place une détection statistique locale de mots sur des colonnes de journaux. Enfin, nous avons montré que l'utilisation conjointe du voisinage par les boîtes englobantes et par le graphe permettait une grande faculté d'expression pour la description des documents.

5.1 Utilisation basique des opérateurs liés au graphe

Le but de cette première application est d'écrire une grammaire, de manière intuitive, en utilisant les opérateurs liés au graphe de voisinage. Nous décrivons donc une formule d'intégrale mathématique (figure 5.1), telle que celle présentée dans le premier chapitre.

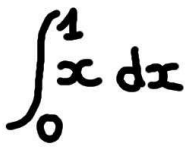


Fig. 5.1 – Formule d'intégrale

5.1.1 Grammaire simple

Comme nous avons vu dans le premier chapitre, une telle formule peut se décrire intuitivement de la manière suivante :

- à gauche du document, un signe intégral ;
- aux extrémités haute et basse de l'intégrale, des bornes d'intégration ;
- à droite de l'intégrale, une expression composée d'une suite de caractères.

Nous traduisons cette description dans le formalisme EPF, en s'appuyant sur les opérateurs liés au graphe. Nous présentons ici le code final ; il est juste simplifié au niveau de la construction des objets reconnus.

La règle de base partie traduit la définition intuitive que nous avons présentée ci-dessus :

```
formule ::=
    AT_ABS(gaucheImage)&&
    integrale ExpInt.
integrale ::=
   composanteIntegrale Int &&
   AT(hautDroite Int) &&
   composanteBorneLiee Int &&
   AT(basDroite Int) &&
   composanteBorneLiee Int &&
   AT(droite Int) &&
   expressionLiee Int.
```

Le symbole intégral ainsi que ses bornes sont déterminés par les terminaux suivants ; on s'assure que la composante de borne est reliée par un lien direct au signe intégral :

```
composanteIntegrale Cmp ::=
   TERM_CMP_GRAPH [noCondC] noCondC lettre Cmp.
composanteBorneLiee Int ::=
   TERM_CMP_GRAPH [(condExisteLienCCGraphe Int)] noCondC lettre Cmp.
```

Une expression est une suite de caractères situés à droite les uns des autres :

```
expressionLiee CmpPrec ::=
   composanteCaractereLie CmpPrec CmpSuiv &&
   AT(aCoteDroite CmpSuiv) &&
   expressionLiee CmpSuiv.
expressionLiee _ . %Cas d'arrêt
```

Pour extraire un caractère (qui est un terminal), on utilise la condition `ccPlusLieeGraphe` qui permet de détecter la composante la plus liée à la celle précédemment trouvée.

```
composanteCaractereLie CmpBase Cmp ::=
   TERM_CMP_GRAPH [(ccPlusLieeGraphe CmpBase)] noCondC lettre Cmp.
```

C'est ce code EPF qui est donc compilé pour produire un analyseur d'intégrales. La figure 5.3 montre un exemple d'analyse pas à pas ainsi que le résultat produit.

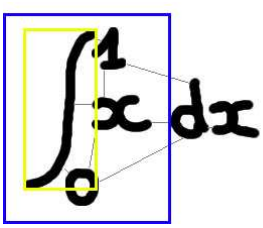
5.1.2 Gestion du bruit

Nous avons, dans un deuxième temps, proposé une variante de cette grammaire de description de intégrales, prenant en compte une éventuelle présence de bruit dans l'image. C'est le cas par exemple de la figure 5.4. Cela nous a permis de valider l'opérateur `FIND_GRAPH` en modifiant la règle principale de la grammaire de la manière suivante :

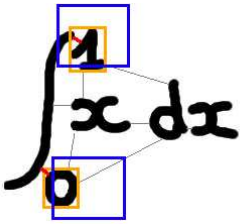
```
formule ::=
    AT_ABS(gaucheImage) &&
    FIND_GRAPH(expressionIntegrale ExpInt) UNTIL(noStopRule).
```

Cela signifie donc que, dans la zone de gauche de l'image, l'analyseur va essayer, par ordre de proximité dans le graphe, d'appliquer la règle `expressionIntegrale`, sur chacun des éléments, jusqu'a ce que l'un d'eux réussisse. Ainsi, sur l'exemple donné, la règle va être essayée plusieurs fois avant de trouver la formule d'intégrale entière.

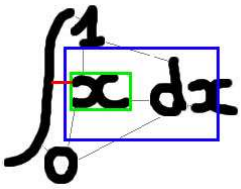
Cette première application nous a permis de valider les opérateurs de base liés au graphe de voisinage.



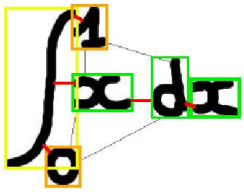
(a) Reconnaissance du signe intégral



(b) Analyse des bornes



(c) Début de l'analyse de l'expression



(d) Résultat final

bleu	les zones de recherche
rouge	les liens du graphe utilisés
jaune	les composantes détectées par `composanteIntegrale`
orange	les composantes détectées par `composanteBorneLiee`
vert	les composantes détectées par `composanteCaractereLie`

FIG. 5.2 – Sémantique des couleurs

FIG. 5.3 – Mécanisme de reconnaissance d'une formule d'intégrale

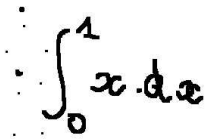


FIG. 5.4 – Formule d'intégrale bruitée

5.2 Utilisation de l'outil statistique local

L'utilisation du graphe de voisinage, basé sur le pavage de Voronoï, dans le cadre de DMOS, permet d'envisager d'extraire les informations du graphe en tenant compte du contexte. C'est ce que nous faisons avec l'outil d'extraction de statistiques présenté au paragraphe 4.3.5.

Pour mettre en application cet outil, nous avons choisi d'effectuer l'analyse de mots et de lignes. Nous présentons donc dans un premier temps la grammaire de description que nous avons mis en place, avant de l'appliquer à des extraits de journaux.

5.2.1 Définition des concepts de ligne et de mots

5.2.1.1 Principe global

Pour définir la grammaire de reconnaissance des mots et des lignes, nous partons de la définition intuitive : un mot est une suite de lettres, une ligne est une suite de mots, les lettres d'un même mot étant plus rapprochées qu'entre deux mots. Nous transposons alors cette définition avec les informations contenues dans le graphe de voisinage :

Mot : suite de composantes liées dans le graphe par un arc de distance inférieure à un seuil `S` ;

Ligne : suite de mots liés dans le graphe par un arc de distance supérieure au seuil `S`.

Le seuil `S` dont il est question ici est celui séparant les distances inter-lettres des distances inter-mots. Il est calculable statistiquement, dans une zone locale, grâce à l'outil présenté au paragraphe 4.3.5.

L'analyse d'une ligne de texte se fait donc en trois étapes :

1. recherche de la zone locale dans laquelle se situe la ligne ;
2. calcul du seuil de distance inter-mots/inter-lettres dans cette zone ;
3. détection des mots et de la ligne grâce au seuil.

La première étape est réalisée classiquement par DMOS, avec les opérateurs de reconnaissance basés sur le graphe ou les boîtes englobantes. Le second point consiste à un appel de l'outil présenté paragraphe 4.3.5 et permet de récupérer un seuil `Seuil`. Ce seuil est extrait grâce à la méthode des k-moyennes et en privilégiant, pour chaque composante de la zone, la distance à ses deux plus proches voisins. Nous détaillons ici l'extrait de grammaire permettant de réaliser la troisième étape.

5.2.1.2 Grammaire utilisée

Nous présentons la grammaire telle que nous l'avons implémentée en EPF ; nous avons simplifié la partie de construction de l'objet finalement reconnu.

La règle principale spécifie qu'une ligne est composée de mots, au moins un, situés à droite les uns des autres. On propage le `Seuil` pour la recherche de chacun des éléments.

```
ligneDeTexte Seuil ::=
    mot Seuil MonMot &&
    AT(droiteMot MonMot) &&
    ligneDeTexte Seuil .
ligneDeTexte _ . %Cas d'arrêt
```

Un mot est défini comme au moins une lettre, suivie d'une fin de mot à la droite de celle-ci.

```
mot Seuil MonMot ::=
    premiereLettreDeMot PremiereCC &&
    AT(droiteCompLigneGraphe PremiereCC) &&
    finDeMot Seuil PremiereCC.
```

Pour détecter la fin d'un mot, on se base toujours sur la composante précédente. Tant que la distance séparant cette composante de la suivante est inférieure au seuil, on reste dans le même mot. Si on ne trouve plus de composante suffisamment proche, c'est la fin du mot.

```
finDeMot Seuil DerniereCC ::=
    lettreDeMot Seuil DerniereCC CetteCC &&
    AT(droiteCompLigneGraphe DerniereCC) &&
    finDeMot Seuil CetteCC.
finDeMot _ _. %Cas d'arrêt
```

Les détecteurs de terminaux permettent de trouver la première lettre d'un mot, sans aucune pré condition. Par contre, la recherche de la lettre suivante d'un mot est soumise à la fois à l'existence d'un lien avec la composante précédente, mais aussi à une distance avec cette composante inférieure au seuil.

```
premiereLettreDeMot CCTrouvee ::=
    TERM_CMP_GRAPH noCondC noCondC compLigne CCTrouvee.
lettreDeMot Seuil CCPrecedente CCTrouvee ::=
    TERM_CMP_GRAPH [(condExisteLienCCGraphe CCPrecedente),
                    (condDistanceLienMax CCPrecedente Seuil)]
                   noCondC compLigne CCTrouvee.
```

5.2.2 Application sur des colonnes de journaux

Nous présentons ici l'application de la grammaire de détection des lignes et des mots à des colonnes de journaux.

5.2.2.1 Documents étudiés

Le but de cette application est de montrer l'intérêt de pouvoir apprendre localement le seuil de distance inter-mots/inter-lettres. En effet, sur les colonnes de journaux étudiées, la taille de la police est variable selon qu'il s'agisse du titre, du sous-titre, de l'auteur ou du corps de texte. Un exemple de document traité est présenté figure 5.5. Il est nécessaire pour pouvoir segmenter correctement les mots, d'apprendre, pour chaque ligne, le seuil adapté.

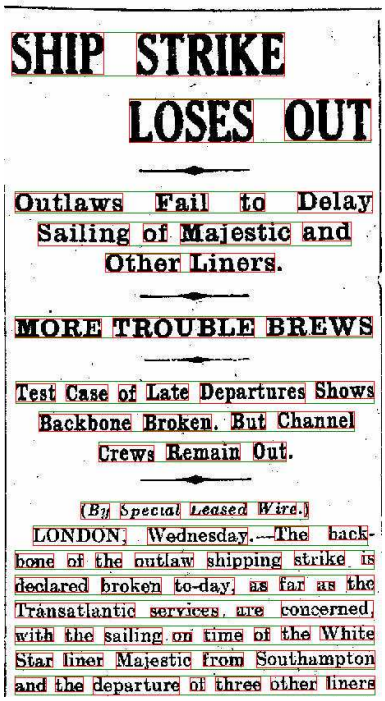
5.2.2.2 Résultats obtenus

Nous avons appliqué la grammaire de reconnaissance des lignes sur une base de 15 documents de colonnes de journaux, composés de 2588 mots à extraire, extraites du *International Herald Tribune*, années 1900, 1925 et 1950. Un des exemples de résultat est visible sur la figure 5.5 où les mots extraits sont représentés en rouge et les lignes en vert.

Nous avons utilisé un outil de statistiques pour évaluer les résultats. Ceux-ci sont visibles sur la figure 5.6.

L'utilisation locale des seuils se montre très efficace puisque plus de 98% des mots à extraire sont reconnus. Les mots trouvés en trop, et ceux manquants, sont liés à un même problème : celui de sur-segmentation. Ceci se produit lorsqu'on tente d'apprendre un seuil sur une ligne ne comprenant qu'un seul mot. Dans ce cas, chacune des lettres est segmentée comme étant un mot. Le résultat est également parfois faussé, pour les mêmes raisons, lorsqu'on détecte le seuil uniquement sur deux mots. C'est la seule erreur que nous ayons dans la méthode d'extraction des mots.

Ceci laisse apercevoir les limites de l'analyse locale : s'il est utile d'apprendre localement les seuils liés au document, il ne faut pas trop restreindre la zone d'analyse, au risque d'avoir



SHIP STRIKE

LOSES OUT

Outlaws Fail to Delay Sailing of Majestic and Other Liners.

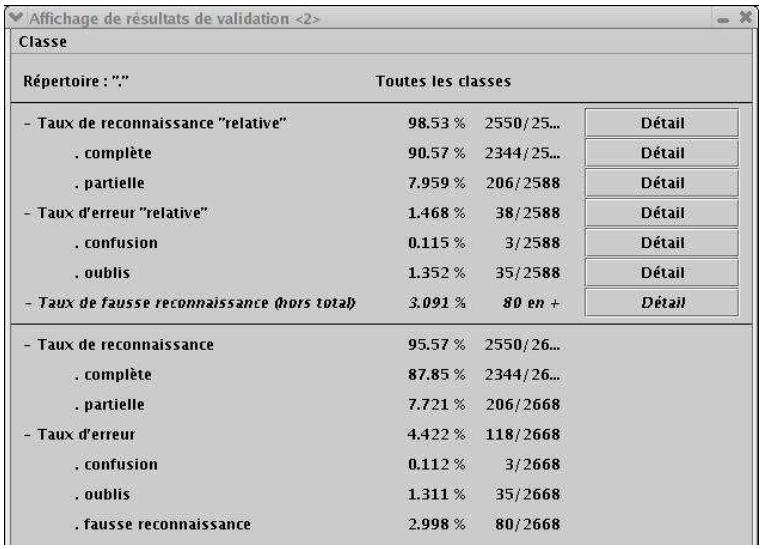
MORE TROUBLE BREWS

Test Case of Late Departures Shows Backbone Broken. But Channel Crews Remain Out.

(By Special Leased Wire.)

LONDON, Wednesday.—The backbone of the outlaw shipping strike is declared broken to-day, as far as the Transatlantic services are concerned, with the sailing on time of the White Star liner Majestic from Southampton and the departure of three other liners

FIG. 5.5 – Exemple de colonne de journal traitée



Affichage de résultats de validation <2>

Classe

Répertoire : "."	Toutes les classes		
- Taux de reconnaissance "relative"	98.53 %	2550/25...	Détail
. complète	90.57 %	2344/25...	Détail
. partielle	7.959 %	206/2588	Détail
- Taux d'erreur "relative"	1.468 %	38/2588	Détail
. confusion	0.115 %	3/2588	Détail
. oublis	1.352 %	35/2588	Détail
- Taux de fausse reconnaissance (hors total)	*3.091 %*	*80 en +*	*Détail*
- Taux de reconnaissance	95.57 %	2550/26...	
. complète	87.85 %	2344/26...	
. partielle	7.721 %	206/2668	
- Taux d'erreur	4.422 %	118/2668	
. confusion	0.112 %	3/2668	
. oublis	1.311 %	35/2668	
. fausse reconnaissance	2.998 %	80/2668	

FIG. 5.6 – Résultats de l'analyse de 15 documents de colonnes de journaux

des résultats erronés. Dans le cas de notre analyse, un apprentissage des seuils pourrait se faire, par exemple, sur tout un paragraphe de texte ayant la même police.

5.3 Utilisation mixte du graphe et des boîtes englobantes

L'intégration des graphes de voisinage que nous avons réalisée dans DMOS permet d'utiliser, selon le souhait de l'utilisateur, soit le voisinage basé sur de boîtes englobantes,

soit celui s'appuyant sur le graphe. Il est également possible d'utiliser conjointement ces deux méthodes, dans le cadre de la description d'un même document. C'est ce que nous proposons de faire pour analyser des décrets de naturalisation manuscrits.

5.3.1 Documents étudiés

Nous étudions des registres de décrets de naturalisation manuscrits de la fin du 19^{eme} siècle.

Une page d'un de ces registres est présentée figure 5.7. Chaque page contient plusieurs actes dont les numéros sont inscrits dans la marge. Pour chaque acte, le corps de texte commence par le nom et le prénom de la personne concernée. Le but de l'analyse est d'isoler les numéros des actes et les noms associés dans la page.

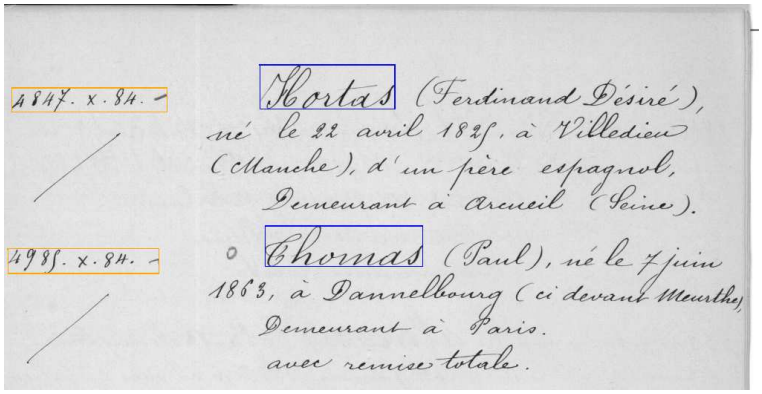


FIG. 5.7 – Exemple d'une page de registre de décret de naturalisation

Une grammaire existante permettait déjà cette reconnaissance ; nous l'avons modifiée et complétée en appliquant les outils liés au graphe lorsque cela semblait plus adapté.

5.3.2 Décomposition de l'analyse

L'extraction des informations contenue dans les décrets se décompose en quatre étapes. Par chacune d'entre elles, nous allons présenter le mécanisme de voisinage que nous avons choisi d'utiliser.

5.3.2.1 Détection des colonnes

La première étape consiste à déterminer la position de la marge du document par rapport au corps de texte. On recherche donc, de manière globale, et sans consommer de terminaux, les alignements du texte afin de récupérer les délimitations de la colonne de marge.

Cette étape de l'analyse se base sur une reconnaissance globale du document. Or, l'intérêt du graphe de voisinage se fait plus sentir pour l'étude locale de composantes liées par un lien direct ou à faible distance. C'est pourquoi nous conservons la phase de recherche des alignements en se basant sur les boîtes englobantes.

La suite de l'analyse va s'appuyer sur ces alignements : on va extraire des couples numéro/nom situés de part et d'autre de la limite de la marge.

5.3.2.2 Détection des numéros

La recherche du numéro est limitée en largeur à la marge détectée précédemment. On détecte une composante connexe susceptible d'appartenir au numéro, puis on va chercher à droite et à gauche de cette composante des éléments alignés à la composante initiale, formant le numéro.

Nous proposons d'utiliser une écriture basée sur le graphe pour cette détection. En effet, il semble naturel d'exprimer un numéro comme un ensemble de composantes connexes liées par un lien direct dans le graphe. Cette définition nous permet de détecter toutes les composantes d'un numéro, en évitant, entre autres, les problèmes de recouvrement de boîtes englobantes.

Ainsi, sur l'exemple présenté figure 5.8, les boîtes englobantes des deux morceaux du *N* se recouvrent totalement en horizontale et la partie de droite n'était pas détectée avec la méthode initiale de voisinage. L'utilisation du graphe permet d'extraire cette composante.

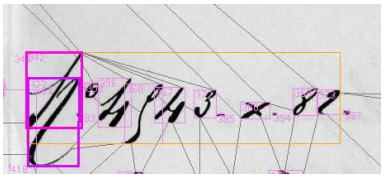


FIG. 5.8 – Exemple de recouvrement de boîtes englobantes dans un numéro

5.3.2.3 Détection des lignes

Afin d'extraire le nom, l'étape suivante consiste à rechercher une ligne, à une position définie par le numéro associé et à droite de la marge. Une ligne de texte est composée d'au moins cinq composantes, dont trois alignées. On s'assure que cette ligne est bien au début d'un paragraphe, c'est à dire qu'elle est suivie d'une autre ligne en dessous, sauf si c'est la fin de la page.

Avec le graphe de voisinage, la détection d'alignements (horizontal ou vertical) d'éléments fonctionne relativement mal puisque l'orientation du lien n'est pas toujours significative de la position relative des objets, et encore moins d'un alignement des composantes. Pour la détection de la ligne avec le graphe, il faudrait donc absolument placer l'analyse dans une zone de recherche limitée en hauteur. D'autre part, on pourrait penser que l'écart entre les lignes est supérieur à l'écart entre les mots, ce qui ne se vérifie pas dans le cas d'écriture manuscrite.

C'est pourquoi on continue à utiliser la méthode des boîtes englobantes pour déterminer grossièrement la position de la ligne.

5.3.2.4 Détection des noms

Une fois la zone de la ligne contenant le nom détectée, il faut extraire les composantes correspondant au nom recherché. Dans la grammaire initiale, entièrement basée sur les boîtes englobantes, on estimait que le nom était situé dans la première moitié de la ligne. On renvoyait donc la liste des composantes situées avant la moitié de la ligne.

Partant du principe qu'en réalité le nom correspondait aux deux premiers mots de la ligne (on préfère extraire un prénom en trop, plutôt que d'oublier la moitié d'un nom

composé), nous avons choisi d'utiliser le graphe de voisinage afin de détecter les mots de la ligne, selon la méthode présentée au paragraphe 5.2.

Disposant de la zone dans laquelle se trouve la ligne, nous appliquons une recherche du seuil de distances inter-mots/inter-lettres dans cette ligne, puis nous détectons les mots. Les deux premiers forment le nom de famille attendu.

Si la méthode semble correcte en théorie, la détection des mots est délicate en pratique. En effet, nous travaillons ici sur du manuscrit, et les distances entre lettres et entre mots sont loin d'êtres régulières. Nous allons le montrer sur deux exemples.

Une erreur courante est liée à la majuscule du nom ou du prénom. En effet, sur ces documents manuscrits, la première lettre est souvent plus détachée du reste du mot. Lors de la détection, elle est donc comptée comme un mot à part.

La figure 5.9 illustre ce problème : la distance entre le R et le e est plutôt de l'ordre de la distance entre le t et la , que de celle entre les deux jambes du n. Le R est donc compté comme un mot à part.

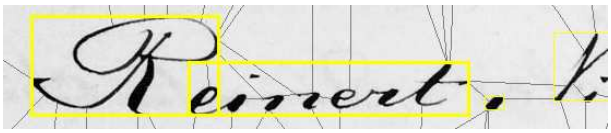


FIG. 5.9 – Exemple de sur-segmentation des mots

Il arrive également que l'espacement entre les mots soit complètement irrégulier, parfois même plus faible qu'entre deux lettres d'un même mot. C'est la cas sur la figure 5.10, où nous avons mis en évidence des segments du même ordre de grandeur, et qui sont alternativement des espaces inter-lettres ou inter-mots.

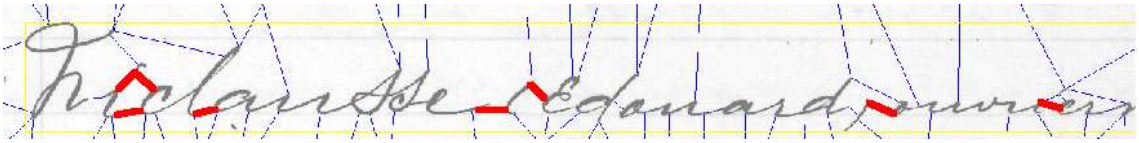


FIG. 5.10 – Exemple d'irrégularité des espaces sur une ligne

Ce problème d'espacement irrégulier entre les lettres limite la reconnaissance de l'écriture manuscrite. Même si le voisinage sur un graphe est beaucoup plus adapté que les boîtes englobantes pour la détection des mots, les irrégularités liées à l'écriture manuscrite rendent difficiles l'extraction dans ce cas.

Cependant, la méthode DMOS offre un avantage certain : la possibilité de combiner les informations extraites du graphe avec des informations liées aux boîtes englobantes. En effet, les boîtes englobantes permettent une analyse globale du document. Lorsqu'il est nécessaire de positionner les éléments les uns par rapport aux autres avec précision, l'utilisation du graphe de voisinage offre une détection plus fine.

5.4 Bilan sur les apports du graphe de voisinage

La validation des outils mis en place dans DMOS nous permet de dresser un bilan sur les apports de l'utilisation conjointe de la méthode générique DMOS et d'un graphe de voisinage basé sur le diagramme de Voronoï.

5.4.1 Spécificité dans l'utilisation des diagrammes de Voronoï

La caractéristique principale de cette exploitation des diagrammes de Voronoï réside dans le fait que DMOS permet une utilisation locale du contexte. Ainsi, nous avons montré l'intérêt d'exploiter le graphe localement pour extraire des statistiques sur des distances inter-mots. Il serait également envisageable de construire des sous graphes afin de profiter d'autres méthodes d'analyse telles que le calcul d'arbre couvrant de poids minimum.

Le graphe de voisinage utilisé comporte plus d'informations qu'un simple diagramme de Voronoï. En effet, on ne se limite pas à la connaissance de l'existence d'un voisinage entre deux composantes, on connaît également la distance, la position du lien ou le taux de liaison entre les éléments. Ces informations offrent des possibilités d'exploitation variées du graphe, qui peuvent être utilisées à la demande, selon les types de documents décrits, mais aussi à l'intérieur de la description d'un même document.

5.4.2 Apports pour DMOS

L'utilisation d'un graphe de voisinage nécessite la manipulation de cet objet tout au long de l'analyse. Nous dressons donc le bilan sur intérêts d'un tel outil.

5.4.2.1 Amélioration de la précision de la recherche dans une zone

Un des grands intérêts de l'utilisation du graphe de voisinage est qu'il permet de positionner localement des composantes avec beaucoup de finesse. En effet, nous avons vu que, une fois la zone de recherche fixée, le graphe intervient pour trier par ordre de distance les éléments dans cette zone. Ceci se fait en prenant en compte les arcs du graphe entre une composante de référence et chacune des composantes, ce qui est beaucoup plus précis qu'un calcul de distance entre un point d'ancrage et une boîte englobante.

Cette notion de recherche dans une zone permet également de passer outre les éventuels recouvrements entre boîtes englobantes, puisque les composantes sont analysées de manière indépendante à cela.

La mise en place de pré conditions spécifiques (existence d'un lien, taux de liaison) permet d'apporter des précisions supplémentaires dans la recherche de composantes, qui s'avèrent pertinentes.

5.4.2.2 Intêret de l'extraction locale des informations statistiques

Le graphe de voisinage permet de fournir des informations statistiques, qui sont ensuite exploitées de manière symbolique par DMOS. Ceci est d'autant plus intéressant que la coopération entre DMOS et le graphe est bi-directionnelle car les informations statistiques fournies par le graphe sont extraites selon le contexte fourni par l'analyseur.

5.4.3 Choix de l'utilisation du graphe de voisinage

Compte tenu des résultats obtenus dans les diverses applications, nous pouvons préconiser l'utilisation du graphe de voisinage dans des cas précis.

Les informations contenues dans le graphe sont liées à une vision locale du document : existence d'une relation de voisinage, distance. C'est pourquoi leur exploitation est très adaptée à une détection de composantes proches les unes des autres et dont on souhaite établir un positionnement relatif précis. L'analyse de documents manuscrits, plus sujette à

des problèmes de recouvrements de boîtes englobantes, bénéficie particulièrement de cette notion de voisinage.

En revanche, dans le cas d'une étude globale d'un document, le graphe de voisinage ne semble pas apporter, pour le moment, d'informations sur le positionnement relatif des objets. Ainsi, dans le cas de la détection d'alignements dans les décrets de naturalisation, nous avons préféré utiliser les boîtes englobantes. Pour pouvoir utiliser le graphe de voisinage de manière globale, il faudrait pouvoir regrouper des composantes (membres de non-terminaux par exemple) entre elles et gérer une hiérarchie de graphes.

Enfin, que cela soit d'un point de vue global ou local, les données statistiques représentent, lorsque cela est nécessaire, des informations utiles à l'analyse des documents.

Conclusion

La méthode DMOS est une méthode générique de reconnaissance de la structure de documents ; elle a déjà été utilisée pour en décrire de nombreux types grâce au formalisme grammatical EPF. A chaque grammaire, on associe l'analyseur associé capable d'effectuer la détection de la structure. Cet analyseur se base initialement sur une notion de voisinage entre boîtes englobantes qui manque dans certains cas de précision. L'objectif de ce stage était de proposer une alternative au mécanisme de recherche des composantes en se basant sur un graphe de voisinage extrait du diagramme de Voronoï.

Nous avons donc mis en oeuvre un graphe de voisinage construit par propagation d'un masque de chanfrein. Nous avons ensuite intégré l'utilisation du graphe à la méthode DMOS. Ceci a permis de compléter la méthode en élargissant les facultés de description. En effet, le graphe permet d'une part une définition plus précise du positionnement relatif de composantes connexes, ce qui a entraîné la création de nouveaux détecteurs de terminaux. D'autre part, le graphe permet d'accéder à des informations statistiques sur les relations entre composantes, telles que les seuils de distance utilisés pour la détection des mots dans une ligne.

Ces travaux ont donné lieu à l'écriture d'un article [13], *Using a neighbourood graph based on Voronoï tessellation with DMOS, a generic method for structured document recognition*, qui sera publié dans le cadre de GREC 2005 (*Sixth IAPR International Workshop on Graphics Recognition*), les 25 et 26 août 2005, à Hong Kong.

Il sera intéressant de continuer à exploiter les informations contenues dans le graphe de voisinage de manière à exprimer des notions de positionnement entre non-terminaux de la grammaire. Par exemple, au delà du positionnement relatif de deux composantes d'un mot, il pourrait être utile de connaître le placement d'un mot par rapport à un autre. Il faudra pour cela mettre en place une hiérarchie de graphe permettant de regrouper, au fur et à mesure de l'analyse, les éléments reconnus.

RÉFÉRENCES

[1] Dominique Attali. Squelettes et graphes de Voronoï 2D et 3D. Thèse, Université de Grenoble I, 1995.

[2] Etienne Bertin. Diagrammes de Voronoï 2D et 3D : Application en analyse d'images. Thèse, Université de Grenoble I, 1994.

[3] Mark Burge and Gladys Monagan. Using the Voronoï tessellation for grouping words and multi-part symbols in documents. *Graphics Recognition : Recent Advances and Perspectives*, pages 38–49, 1997.

[4] Bertrand Coüasnon. DMOS : A generic document recognition method to application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. *International Conference on Document Analysis. (ICDAR'01)*, pages 215–220, 2001.

[5] Bertrand Coüasnon. Dealing with noise in DMOS, a generic method for structured document recognition : an example on a complete grammar. *Graphics Recognition : Recent Advances and Perspectives*, pages 38–49, 2004.

[6] Pascal Garcia. *Reconnaissance de formules mathématiques.* Rapport de DEA, Université de Rennes I, 2000.

[7] Pascal Garcia and Bertrand Coüasnon. Using a generic document recognition method for mathematical formulae recognition. *GREC, IAPR International Workshop on Graphic Recognition*, pages 236–244, 2001.

[8] A. Gribov and E. Bodansky. Vectorization with the Voronoï L-diagram. *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 1015, 2003.

[9] Rangachar Kasturi, Lawrence O'Gorman, and Venu Govindaraju. Document image analysis : a primer. *Sandhana*, pages 3–22, 2002.

[10] Kise Koichi, Motoi Iwata, and Keinosuke Matsumoto. On the application of Voronoï diagrams to page segmentation. *Document Layout Interpretation and its Application (DLIA'99)*, 1999.

[11] Kise Koichi, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area Voronoï diagram. *Computer Vision and Image Understanding*, pages 370–382, 1998.

[12] Kise Koichi, Akinori Sato, and Keinosuke Matsumoto. Document image segmentation as selection of Voronoï edges. *Workshop on Document Image Analysis (DIA'97)*, pages 32–39, 1997.

[13] Aurélie Lemaitre, Bertrand Coüasnon, and Ivan Leplumey. Using a neighbourood graph based on Voronoï tessellation with DMOS, a generic method for structured document recognition. *GREC, IAPR International Workshop on Graphic Recognition*, 2005.

[14] Ivan Leplumey and Charles Quéguiner. Un graphe de voisinage basé sur l'utilisation des distances discrètes. *Conférence Internationale Francophone sur l'Ecrit et le Document (CIFED'2002)*, pages 41–50, 2002.

[15] Yue Lu, Zhe Wang, and Chew Lim Tan. Word extraction using area Voronoï diagram. *Conference of Computer Vision and Pattern Recognition (CVRP'03)*, 2003.

[16] Yue Lu, Zhe Wang, and Chew Lim Tan. Word grouping in document images based on Voronoï tessellation. *Workshop on Document Analysis Systems (DAS'04)*, 2004.

[17] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms : A literature survey. *Proc. SPIE Electronic Imaging*, pages 197–207, 2003.

[18] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations : Concepts and Applications of Voronoï Diagrams*. Wiley, 2000.

[19] Edouard Thiel. Les distances de chanfrein en analyse d'images : fondements et applications. Thèse, Université de Grenoble I, 1994.

[20] Edouard Thiel. Géométrie des distances de chanfrein. Habilitation à diriger des recherches, Université de la Méditerranée (Aix-Marseille II), 2001.