Acquisition SVG de vérités terrain
Application aux livres anciens imprimés

Yannick Labarre et Jérémy Selier

Projet de recherche M1 IMA
Département Informatique de l'Université de La Rochelle
Laboratoire L3i, Pôle Sciences et Technologie
Encadrement : Mathieu Delalandre/ Nicholas Journet

Table des matières

1 – Introduction générale

Ce projet concerne l'analyse d'image de document appliquée en particulier aux documents graphiques [1] (cartes, symboles, logos, etc.). Plus spécialement, nous nous intéressons aux parties graphiques présentes dans les documents du patrimoine et entre autres les livres anciens imprimés du XV° - XVIII° siècle. La Figure 1 suivante en donne quelques exemples. En effet, ces livres anciens imprimés sont majoritairement composés de caractères typographiques mais aussi de nombreuses parties graphiques comme les bandeaux, les illustrations, les lettrines et les cul-de-lampes.

Figure 1 : Parties graphiques dans les livres anciens imprimés

Le sujet qui nous intéresse ici est l'évaluation de performances des systèmes d'indexation appliqués à la recherche de ces parties graphiques [2]. L'évaluation de performances est un domaine de recherche émergent depuis les années 90 [3]. Elle aborde les problématiques de constitution de corpus de test et de définition de métriques pour l'évaluation des capacités d'indexation et/ou de reconnaissance des systèmes. Dans le cadre de notre projet nous nous limiterons aux aspects constitution de corpus de test. Pour ce faire deux approches sont communément utilisées dans la littérature [4]. La première est la génération automatique de bases de documents synthétiques à partir de modèles préalablement établis. La seconde elle consiste à demander à un opérateur humain de définir, à partir d'images scannées, la description à produire par un (ou des) système(s) donné(s). On parle alors de vérités terrain pour qualifier ces descriptions.

Dans le cadre de notre travail la spécificité des documents anciens (dégradation, variabilité, complexité, etc.) rend difficile la modélisation des parties graphiques existantes. Une approche de type acquisition de vérités terrain semble donc s'imposer pour l'évaluation de performances des systèmes. Cependant celle-ci soulève deux principales problématiques [4] :

1. Elle est effectuée par des opérateurs humains.
2. Les vérités terrain sont par nature ambiguës à définir.

De façon à palier ces problèmes l'usage habituel est alors de recourir à différents opérateurs afin de faire converger leurs acquisitions en une vérité communément admise [4]. Notre travail s'inscrit dans cette démarche. Nous proposons ici un système permettant l'analyse de zones d'intérêt produites par différents opérateurs. La Figure 2 suivante en illustre le principe.

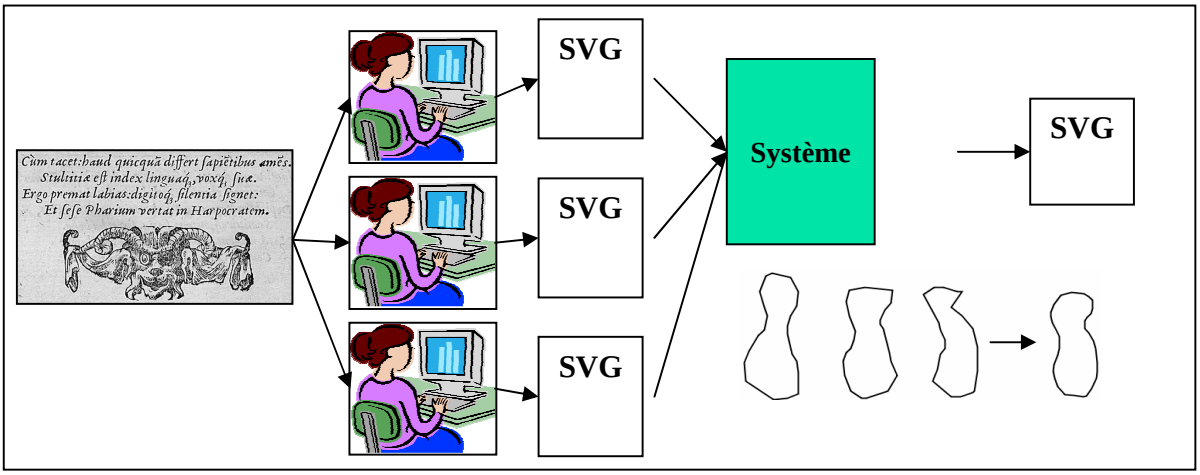


Figure 2 : schéma explicatif de notre approche

Nos zones d'intérêt correspondent à des polygones définis par les opérateurs à partir d'éditeurs SVG. Notre système analyse alors différents fichiers SVG dans le but de les fusionner. Ce processus implique différentes étapes : parsing, mapping, tri logique et moyennage.

Dans la suite de ce rapport nous présentons tout d'abord dans la section 2 le format SVG. Dans les sections 3 et 4 nous présentons notre système puis ses limites de fonctionnement. Finalement dans la section 5 nous concluons et donnons nos perspectives sur ce travail.

2 – Le format SVG

L'acronyme SVG [5] signifie *Scalable Vector Graphic*, que nous pourrions traduire librement en français par *Graphique Vectoriel Extensible* ou encore *Image Vectorielle Extensible*. Ce format est une norme du W3C (World Wide Web Consortium). Il s'agit d'un sous langage XML [6] (Extensible Markup Language ou Langage de Balisage Extensible). Le format SVG permet de représenter des dessins vectoriels. Un dessin vectoriel est une image stockée sous la forme d'objets définis par des coordonnées mathématiques (lignes, rectangles, cercles, ellipses, courbes, etc.). La Figure 3 suivante donne un exemple de code SVG et sa représentation graphique associée.

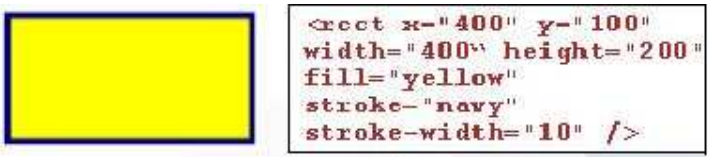


Figure 3 : Exemple de code SVG

Contrairement aux images en mode point, les dessins SVG n'ont pas de taille fixe : ils sont définis par assemblage d'objets de types divers. La taille et l'assemblage de ces objets peuvent être paramétrés. C'est cette propriété qui rend le langage SVG *Scalable* : le dessin vectoriel est extensible. De plus le format SVG propose diverses fonctionnalités comme par exemple des transformations géométriques, les animations, la superposition d'images, etc. Toutes ces caractéristiques rendent aujourd'hui le format SVG populaire [5], de nombreux éditeurs sont désormais disponibles et il est utilisé dans différents domaines. La Figure 4 suivante donne quelques exemples de document SVG :

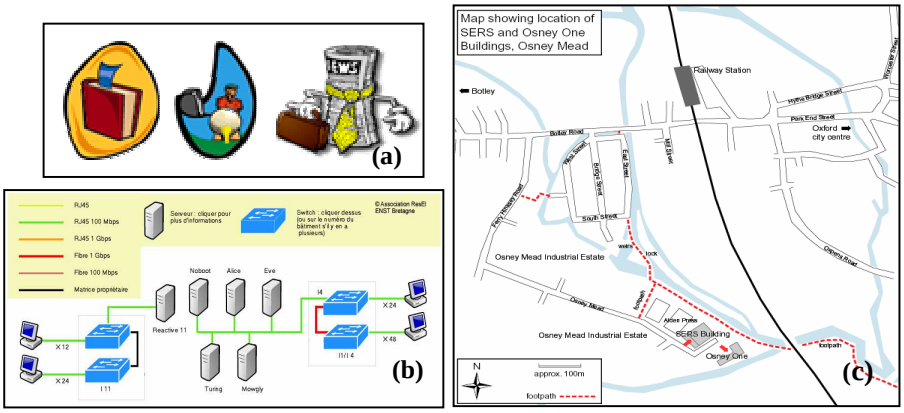


Figure 4 : Exemples de documents SVG
(a) clipart (b) plan de réseau (c) carte géographique

3 – Système d'acquisition de vérités terrain

3.1 – Introduction

Dans cette section nous présentons notre système d'acquisition de vérités terrain, la Figure 5 suivante en donne l'architecture générale. Tout d'abord, à partir d'une même image de document plusieurs opérateurs procèdent à la définition de zones d'intérêt via des éditeurs SVG. Les fichiers générés servent alors de données d'entrée de notre système. Celui-ci les traite en quatre étapes successives : parsing, mapping, tri logique et moyennage.

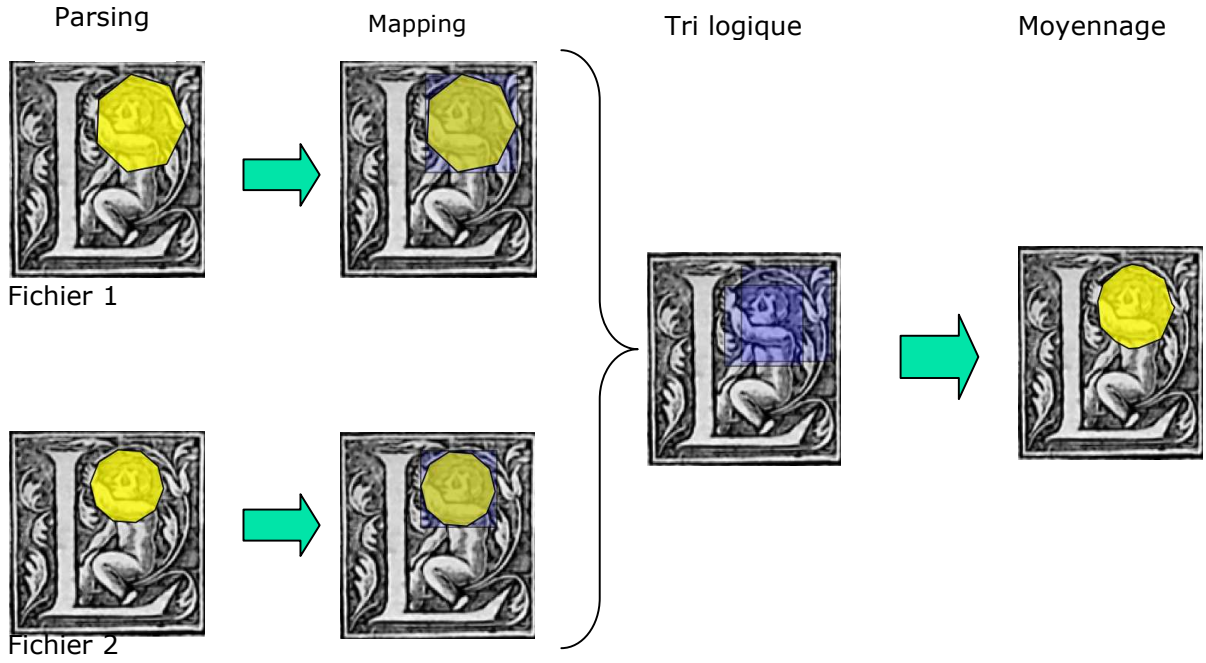


Figure 5 : Architecture de notre système

Le Parsing traite les fichiers SVG afin d'extraire les données correspondantes aux zones d'intérêt définies par les utilisateurs. Il s'agit donc d'une étape de chargement des différentes vérités terrain au sein de notre système.

Le Mapping a pour but d'identifier les recouvrements existants entre les zones d'intérêt définies par les opérateurs. Pour chaque zone plusieurs Mapping sont envisageables selon les recouvrements existants avec les autres zones. Nous avons supervisé notre algorithme de Mapping par un algorithme de tri logique. Ce dernier a alors pour but de sélectionner parmi les Mapping possibles les plus pertinents.

Dans une dernière étape nous procédons au moyennage des zones d'intérêt. Le but est alors de faire converger les différentes zones en une zone commune correspondant au regroupement des différentes vérités.

Nous détaillons chacune de ces quatre étapes dans les sous-sections suivantes 3.2, 3.3, 3.4 et 3.5.

3.2 – Parsing

La première étape de notre système consiste, une fois l'édition des fichiers SVG effectuée, de procéder à leur parsing. Le but est alors de charger les vérités terrain stockées dans ces fichiers au sein de notre système, la Figure 6 en illustre le principe.

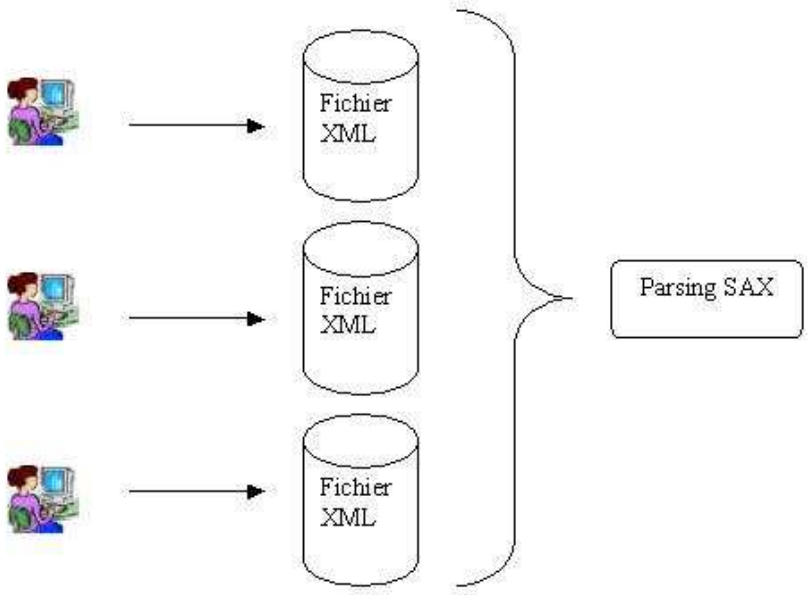


Figure 6 : Schéma explicatif du module du parsing

Le parsing consiste, pour une application informatique donnée, à extraire des informations à partir de fichiers XML. Pour ce faire les applications mettent en oeuvre des parsers, ceux-ci sont de deux types [6] : SAX et DOM. Dans notre cas nous avons utilisé un parser de type SAX (Simple API for XML). En effet, le parsing DOM (Document Object Model) est généralement mis en oeuvre dans des buts d'édition des documents XML. Les documents XML sont alors représentés sous la forme d'arbres chargés intégralement au sein des applications. Le parsing SAX plus léger repose lui sur un modèle événementiel. Le parser parcourt le fichier XML à la recherche de balises spécifiques définies par l'application, et cela sans prise en compte du modèle global du document XML. Pour cela un parser SAX repose sur l'utilisation de trois fonctions principales :

- public void startDocument():appelée au commencement du document.
- public void endDocument(): appelée à la fin du document.
- public void startElement (String name, AttributeList attrs) : appelée lors de la lecture d'une balise ouvrante.

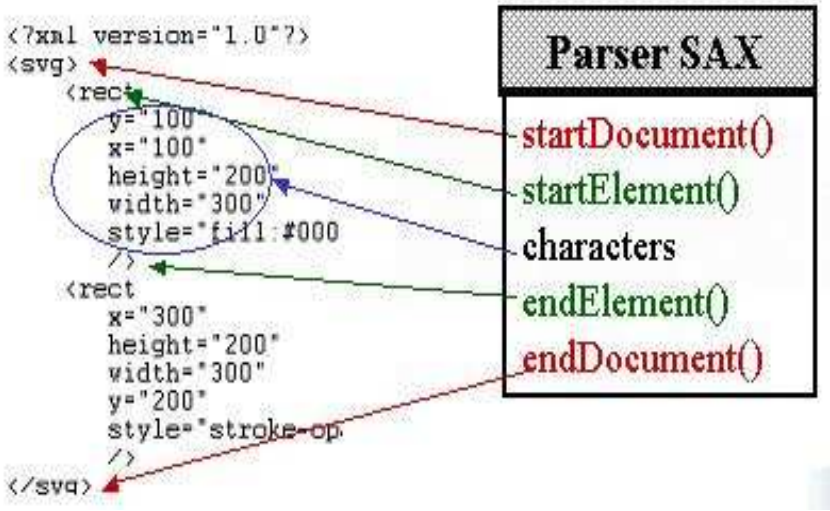


Figure 7 : Explication du parsing SAX

Basé sur l'utilisation d'un parser SAX, notre algorithme de parsing est le suivant :

Entrée : un fichier SVG
Sortie : une liste de polygones avec ses coordonnées
Début
| Tant que le fichier SVG n'est pas entièrement parcourue faire (start_document())
| | Si le balise *polygon* ou *path* détecté (start_element())
| | | Tant qu'il y a des coordonnées a extraire faire (parsePoly() ou parsePath())
| | | | Stocker coordonnée dans un tableau
| | | Fin Tant que
| | | Stocker la liste des coordonnées dans un tableau du i eme *polygon*
| | Fin si
| Fin Tant que (end_document())
Fin

Cet algorithme parse les fichiers SVG à la recherche des balises *Polygon* et *Path*. En effet, les zones d'intérêt se définissent naturellement sous la forme de balises *Polygon*. Néamoins, la balise Path étant générique en SVG certains éditeurs l'utilisent par défaut pour l'enregistrement des objets SVG. La Figure 8 suivante illustre les différences d'écriture entre ces deux balises.

```
<path d="M 220.8835,704.85753 L 209.96678,752.55536 L
149.41208,713.25004 L 100.41767,610.35209 z "/>

<polygon points = "220.8835,704.85753 209.96678,752.55536
149.41208,713.25004 100.41767,610.35209"/>
```

Figure 8 : Deux écritures possibles des polygones en SVG

Une fois les polygones détectés notre algorithme met en œuvre deux fonctions pour leur analyse : parsePoly() et parsePath(). Ces fonctions retournent alors des données décrivant les polygones ajoutés à notre structure de données globale. Cette dernière est présentée dans la Figure 9 suivante. Chacun des fichiers SVG contient une liste de polygone et chaque polygone est composé d'une liste de points (x, y) correspondant aux cotés des polygones.

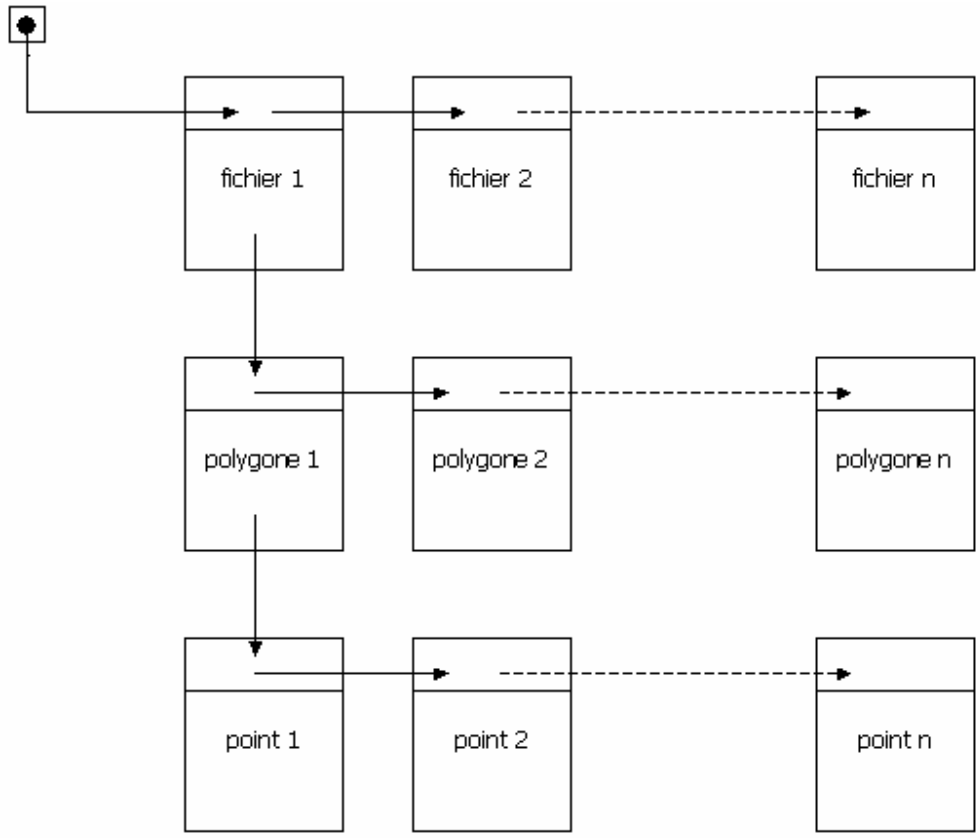


Figure 9 : Structure de données du système

3.3 – Mapping

Nous procédons dans une seconde étapes au mapping des polygones extraits des fichiers SVG. Pour ce faire, nous nous basons sur l'analyse des boites englobantes des polygones de façon à déterminer des indices de recouvrement. Notre algorithme général est présenté ci-dessous. Il se décompose en deux étapes principales : la construction des boites englobantes et leur appariement (Matching).

```
Nom : mapping
Entrées : rien
Sortie : rien
Pour i allant de 0 au nombre de fichiers SVG
    Pour j allant de 0 au nombre de polygones dans le fichier i
        On récupère la boite englobante associée au polygone j du fichier i (R1)
        Pour k allant de i+1 au nombre de fichiers SVG
            Pour l allant de 0 au nombre de polygones dans le fichier k
                On récupère la boite englobante associée au polygone l du fichier k (R2)
                Matching(R1, R2) retourne le taux de recouvrement des 2 rectangles
                Si le matching ne retourne pas –1 Alors
                    On augmente le nombre d'utilisation de 1 pour les 2 rectangles
                    On ajoute les 2 rectangles ainsi que le taux
                Fin Si
            Fin Pour
        Fin Pour
    Fin pour
Fin pour
```

Pour la construction d'une boite englobante nous parcourons tout d'abord tous les sommets du polygone considéré. Nous cherchons ensuite les abscisses et ordonnées de valeurs supérieures et inférieures tel que l'illustre la Figure 10 suivante. A partir de ces valeurs les paramètres (x0,y0, dx, dy) de la boite englobante se déterminent aisément.

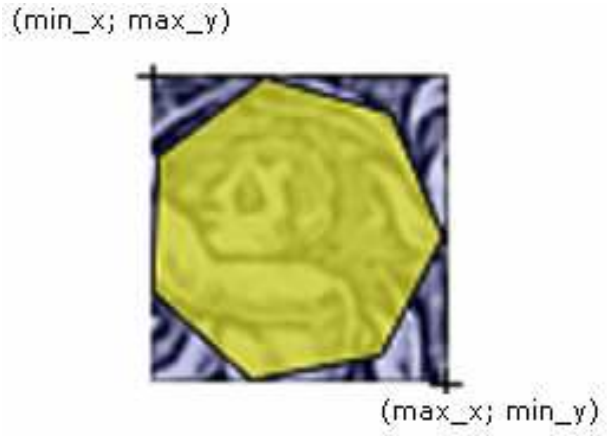


Figure 10 : Calcul de la boîte englobante

A partir des boites englobantes le but est ensuite de déterminer quelles zones d'intérêt se recouvrent. Pour ce faire nous utilisons un algorithme d'appariement des boites englobantes. Différents cas de figure sont envisageables comme l'illustre la Figure 11 suivante.

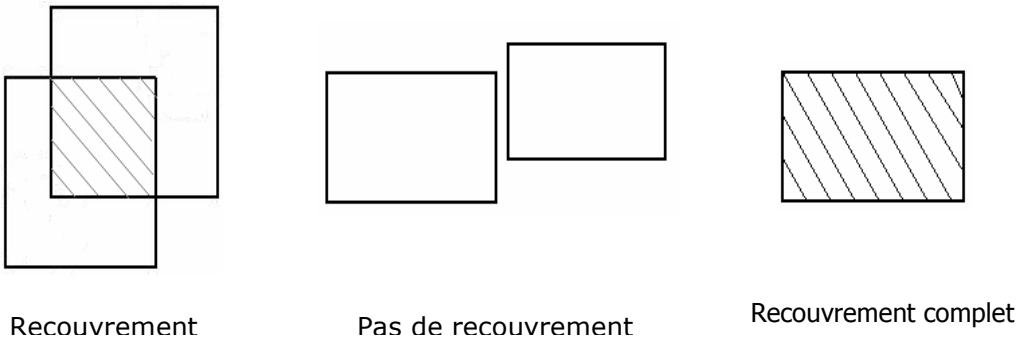


Figure 11 : Cas de recouvrement

Notre algorithme d'appariement est présenté ci-dessous. Celui-ci procède en deux étapes principales : calcul de la zone de recouvrement et calcul du taux de recouvrement entre deux boites englobantes.

```
Nom : matching
Entrées : Rectangle R1, rectangle R2
Sortie : le taux de recouvrement sinon –1
Si il y a une zone commune avec l'intersection de R1 et R2 Alors
|       On calcule l'aire de R1 et de R2 ainsi que de la zone commune
|       On calcule le taux de recouvrement grâce aux airs précédemment calculées
|       On retourne ce taux
Sinon
|       On retourne -1
Fin Si
```

Lorsqu'un recouvrement est détecté[1] entre deux boites englobantes nous procédons alors au calcul du taux de recouvrement. Ce taux s'obtient par application de l'équation suivante :

$$taux = \frac{air\_commune}{(air\_rectangle1 + air\_rectangle2) - air\_commune}$$

3.4 – Tri logique

Le taux de recouvrement entre deux boites englobantes n'est pas toujours suffisant pour déterminer le mapping optimal. La figure suivante illustre ce cas de figure. Sur cet exemple les boites englobantes verte et bleue sont mappées. Cette décision laisse alors les autres zones d'intérêt sans possibilités de Mapping.

[1] Cette détection repose sur l'utilisation du package Java **`java.awt`**

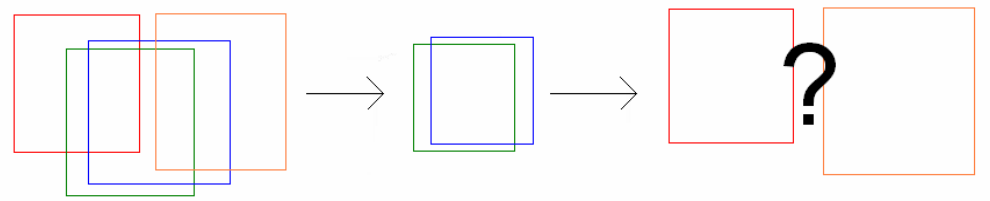


Figure 12 : Mapping par sélection du meilleur taux de recouvrement

De façon à résoudre à ce cas de figure nous utilisons en seconde étape un algorithme de tri logique des solutions de Mapping. Ce tri repose tout d'abord sur l'analyse des taux de recouvrement des boites englobantes. Il repose également sur leur nombre d'utilisation. Ce dernier correspond aux recouvrements possibles pour une boite donnée. En effet, nous le calculons durant le Mapping parallèlement au calcul des taux de recouvrement. Notre algorithme de tri logique a alors pour but de classer les Mapping possibles selon les critères de taux de recouvrement et de nombre d'utilisation. Nous le présentons ci-dessous, il procède de la façon suivante.

```
Nom : trie
Entrée : tableau des matching polygones (matching_poly)
Sortie : tableau trié (T)
Début
|       Trier matching_poly par le taux de recouvrement
|       Pour i=0 a tout les les couples de polygones de matching_poly faire
|       |       On recupere le nombre d'utilisation des 2 polygones de matching_poly[i]
|       |       Si leur nombre d'utilisation n'est pas de –1 alors
|       |       |       not_find mit a faux
|       |       |       Pour j=i+1 a tout les les couples de polygones de matching_poly faire
|       |       |       |       On recupere le nombre d'utilisation des 2 polygones de
|       |       |       |       matching_poly[j]
|       |       |       |       Si leur nombre d'utilisation n'est pas de –1 alors
|       |       |       |       |       On regarde à l'aide du nombre d'utilisation les couples de
|       |       |       |       |       polygone de matching_poly[i]
|       |       |       |       |       par rapport aux couples de polygones de
|       |       |       |       |       matching_poly[j]
|       |       |       |       |       Si prendre le couple de polygones de matching_poly[i]
|       |       |       |       |       |       gêne pour la suite alors
|       |       |       |       |       |       |       not_find mit a vrai
|       |       |       |       |       |       Fin si
|       |       |       |       |       Fin si
|       |       |       |       Fin pour
|       |       |       |       Si not_find est faux alors
|       |       |       |       |       On ajoute dans le tableau trié (T) le couple matching_poly[i]
|       |       |       |       |       Mettre leur nombre d'utilisation a -1
|       |       |       |       Fin si
|       |       |       Fin si
|       |       Fin si
|       Fin pour
Fin
```

Nous détaillons ici le fonctionnement de cet algorithme au travers d'un exemple :

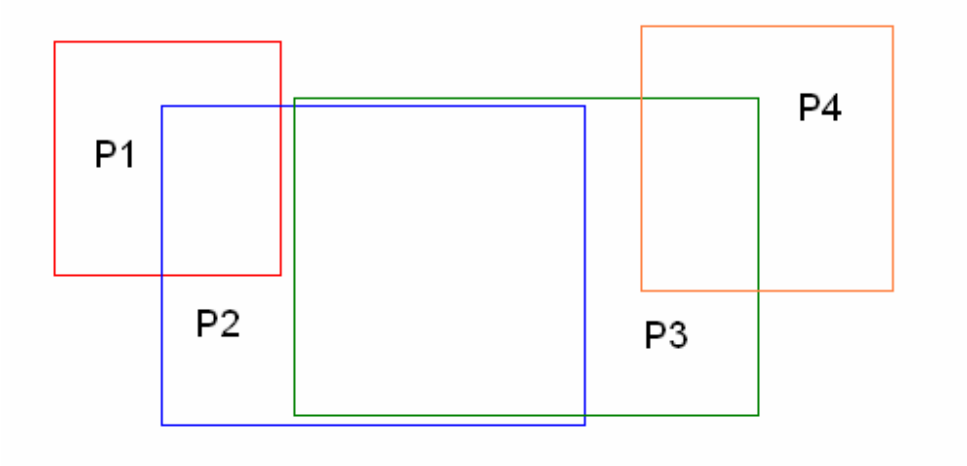


Figure 13 : Cas d'usage de tri

Polygone	Nombres d'utilisation	Taux de recouvrement

Etape 1 (liste des couples triés) :

Polygone 2	2	80%
Polygone 3	2	

Polygone 1	1	40%
Polygone 2	2	

Polygone 3	2	40%
Polygone 4	1	

Etape 2 (on sélectionne celui avec le taux le plus grand) :

Polygone 2	2	80%
Polygone 3	2	

Polygone 1	1	40%
Polygone 2	2	

Polygone 3	2	40%
Polygone 4	1	

Etape 3 (on parcourt le reste des couples) :

Polygone 2	2	80%
Polygone 3	2	

Polygone 1	1	40%
Polygone 2	2	

Polygone 3	2	40%
Polygone 4	1	

Etape 4 :

Polygone 2 présent dans le bloc vert et rouge, on retire alors 1 au nombre d'utilisation dans le bloc vert pour le polygone non présent dans le bloc rouge.

Etape 4 :

Polygone 2	2	80%
Polygone 3	2	

Polygone 1	**0**	40%
Polygone 2	2	

Polygone 3	2	40%
Polygone 4	1	

Etape 5 :

Cela signifie que si l'on prend le bloc rouge, on aura au final un polygone seul, ici le polygone 1, on décide donc de ne pas prendre le bloc rouge et on le retire de la liste. Une fois retiré, on met à jour les nombres d'utilisation.

Polygone 1	1	40%
Polygone 2	**1**	

Polygone 3	**1**	40%
Polygone 4	1	

Etape 6 (on sélectionne celui avec le taux le plus grand) :

Polygone 1	1	40%
Polygone 2	1	

Polygone 3	1	40%
Polygone 4	1	

Etape 7 (on parcourt le reste des couples) :

Polygone 1	1	40%
Polygone 2	1	

Polygone 3	1	40%
Polygone 4	1	

Etape 8 :

Pas de polygone rouge présent dans le bloc vert. On a parcourus la liste des polygones jusqu'au bout sans descendre un nombre d'utilisation à 0, on peut donc sélectionner le bloc rouge.

Etape 9 :

Polygone 1	1	40%
Polygone 2	1	

Polygone 3	1	40%
Polygone 4	1	

Etape 10 :

Et on continue pour retrouver au final :

Polygone 1	1	40%
Polygone 2	1	

Polygone 3	1	40%
Polygone 4	1	

3.5 – Moyennage

Dans une dernière étape nous procédons au moyennage des zones d'intérêt mappées. Pour cela nous utilisons une technique dite de calcul des points moyens. La Figure 14 en illustre le principe. Dans un premier temps les sommets de chacun des polygones mappés sont parcourus tour à tour. Pour chacun de ces sommets on identifie alors l'arrête la plus proche dans le polygone voisin. Le point moyen est obtenu alors par projection orthogonale du sommet sur l'arête voisine. Quand tous les points moyens ont été calculés ils sont ensuite reliés pour construire le polygone moyen.

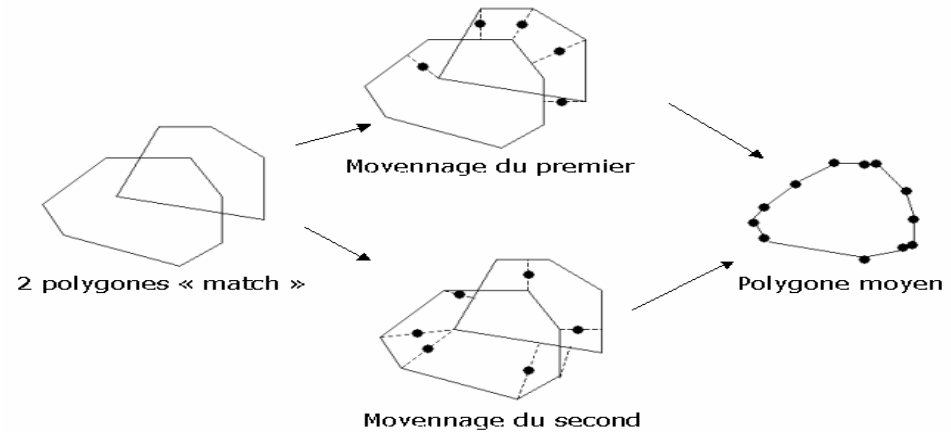
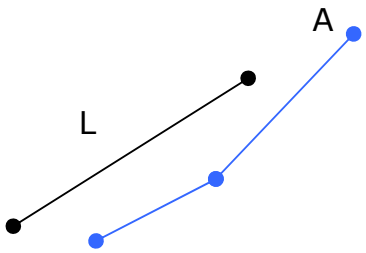


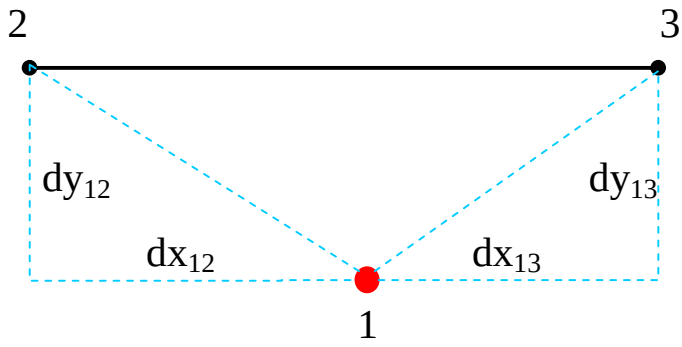
Figure 14 : Principe du moyennage

Dans une première étape notre algorithme recherche pour chacun des sommets l'arrête la plus proche dans le polygone voisin. Cette recherche s'effectue seulement à partir des arrêtes recouvrant ce sommet. La Figure 15 illustre ce principe de recouvrement (overlapping) entre un sommet et une arrête. La Figure 16 détail le calcul que nous mettons en oeuvre pour déterminer ce recouvrement.



Ici, seul le sommet A n'est pas recouvert par l'arrête L

Figure 15 : Principe du recouvrement



1 est recouvert si

$$dx_{12} \times dx_{13} < 0$$

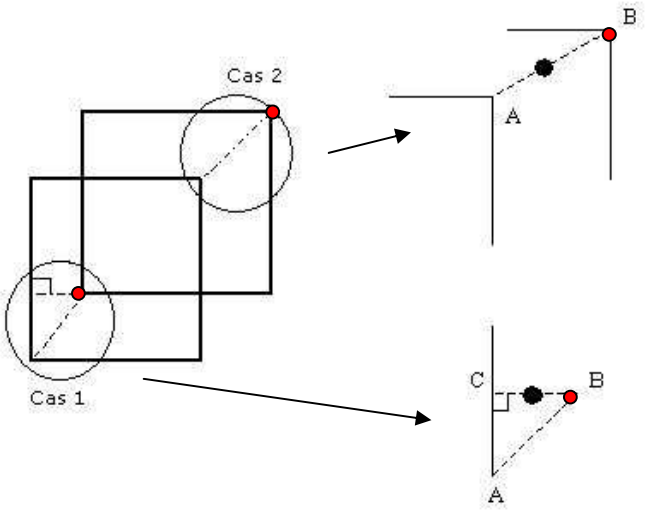
or

$$dy_{12} \times dy_{13} < 0$$

Figure 16 : Test de recouvrement entre un sommet et une arrête

Selon qu'un sommet soit recouvert par une arrête ou non deux calculs se présentent, nous les illustrons sur la Figure 17 :

- Cas 1 : Si le sommet (B) est recouvert par une arrête il est projeté orthogonalement dessus. Le milieu de la droite de projection [BC] correspond alors au point moyen.
- Cas 2 : Dans le cas ou le sommet (B) n'est recouvert par aucune arrête nous calculons le point médian avec le sommet le plus proche du polygone voisin.



Cas 2 : (pas d'overlapping) Prendre le point le plus près.

Cas 1 : (overlapping) Appliquer la projection orthogonale.

Figure 17 : Calculs des points moyens

Une fois tous les points moyens calculés, nous obtenons une série de points non ordonnés. Il est donc nécessaire de les trier afin de reconstituer le polygone moyen. Nous utiliserons pour cela un algorithme de tri basé sur le calcul de la distance Euclidienne entre point. La Figure 18 suivante en illustre le principe.

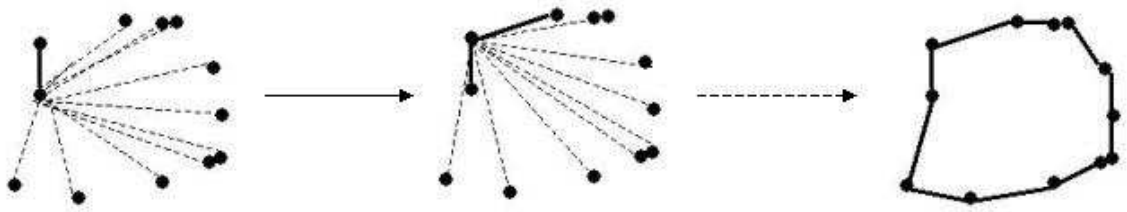


Figure 18 : Principe de construction du polygone moyen

4 – Limites de fonctionnement

Dans la section 3 précédente nous avons présenté le fonctionnement général de notre système d'acquisition de vérités terrain. Bien qu'opérationnel ce système est contraint aujourd'hui à certaines limites de fonctionnement, nous les présentons dans cette section. Ces limites sont de deux natures : le mapping des formes imbriquées et le moyennage des formes concaves.

Lors du matching nous utilisons les boites englobantes des zones d'intérêt pour déterminer leur taux recouvrement. Ce type de méthode présente une limite dans le cas des formes imbriquées, la Figure 19 l'illustre. Dans cet exemple deux zones d'intérêt présentent un taux de recouvrement non négligeable au regard de leurs boites englobantes. Cependant les zones d'intérêt ne présentent elles aucun recouvrement effectif.

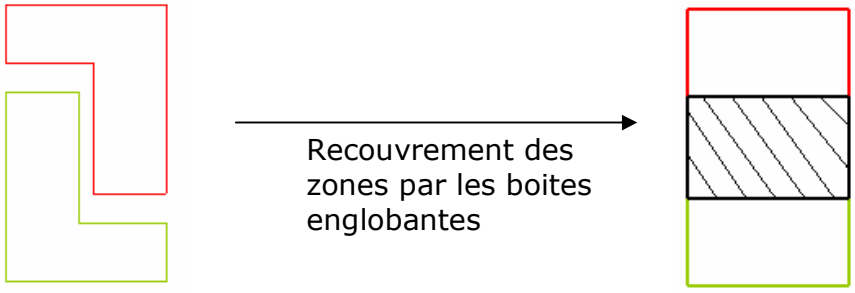


Figure 19 : Problème du mapping par boite englobante des zones imbriquées

Notre seconde limite de fonctionnement concerne le moyennage des formes concaves. En effet, notre tri des points moyens sur critère de distance Euclidienne est mis en défaut dans ce cas. La concavité de zones d'intérêt mappées introduit des proximités entre points moyens non consécutifs dans le polygone à construire. La Figure 20 illustre ce phénomène.

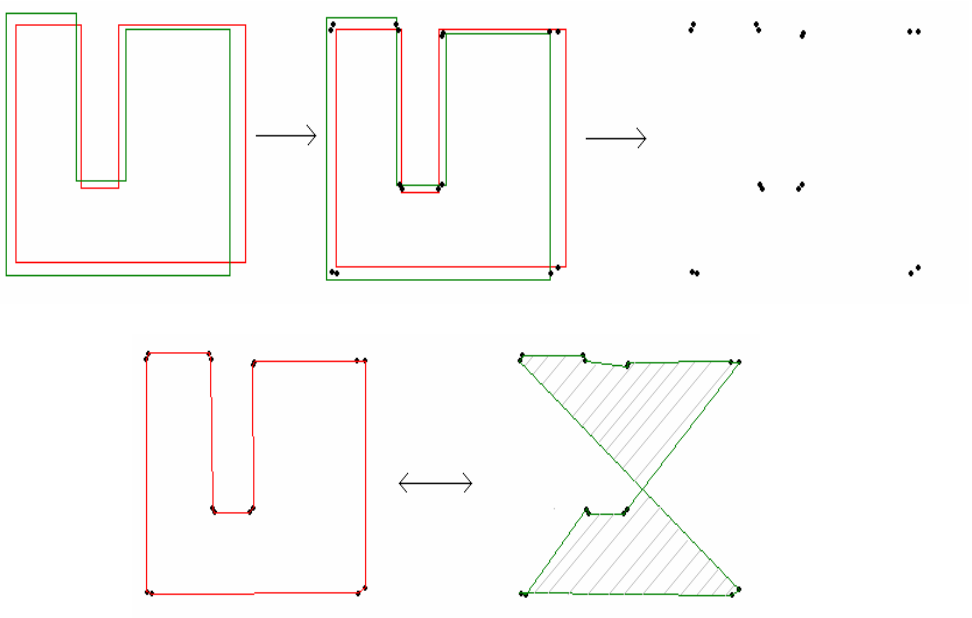


Figure 20 : Problème du moyennage des formes concaves

5 – Conclusion et perspectives

Ce rapport concerne l'évaluation de performances des systèmes d'indexation appliqués à la recherche des parties graphiques. Nous y avons présenté un système d'acquisition de vérités terrain. Celui-ci permet l'analyse de zones d'intérêt produites par différents opérateurs. Ces zones d'intérêt correspondent à des polygones définis à partir d'éditeurs SVG. Notre système analyse alors les différents fichiers SVG dans le but de les fusionner. Le but est alors de produire une vérité commune à partir des différentes éditions. Ce processus implique plusieurs étapes : parsing, mapping, tri logique et moyennage. Ce système est aujourd'hui opérationnel, notre approche est cependant soumise à différentes limites de fonctionnement. En effet, elle est particulièrement sensible au traitement des formes imbriquées et concaves. De façon à résoudre ces problèmes une perspective d'amélioration de ce travail consisterait à exploiter une représentation en graphe d'adjacence des régions [7].

6 – Références

[1] S. Ablameyko and T. Pridmore. *Machine Interpretation of Line Drawing Images*. Springer Verlag Publisher, 2000.

[2] R. Pareti and al. On defining signatures for the retrieval and the classification of graphical dropcaps. In *Conference on Document Image Analysis for Libraries (DIAL)*, pages 220-231, 2006.

[3] I. Phillips and A. Chhabra*. Empirical performance evaluation of graphics recognition systems*. Pattern Analysis and Machine Intelligence (PAMI), 21(9) : 849-870, 1999.

[4] D. Lopresti and G. Nagy. *Issues in ground-truthing graphic documents*. In Workshop on Graphics Recognition (GREC), volume 2390 of Lecture Notes in Computer Science (LNCS), pages 46-66, 2002.

[5] D. Duce, I. Herman, and B. Hopgood. *SVG Tutorial*. Oxford Brookes University and World Wide Web Consortium, 2002.

[6] A. Michard. *XML Langage et Application*. Editions Eyrolles, 2 edition, 2000.

[7] K. Zouba. Un algorithme pour la construction de graphes de régions à partir de graphiques vectoriels. Mémoire de Master, Laboratoire PSI, Université de Rouen, France, 2005.

Résumé : Ce projet concerne l'évaluation de performances des systèmes d'indexation appliqués à la recherche des parties graphiques. Plus particulièrement, nous nous sommes intéressés aux problèmes d'acquisition de vérités terrain. Nous proposons ici un système permettant l'analyse de zones d'intérêt produites par différents opérateurs. Ces zones d'intérêt correspondent à des polygones définis à partir d'éditeurs SVG. Notre système analyse alors les différents fichiers SVG dans le but de les fusionner. Le but est alors de produire une vérité commune à partir des différentes éditions. Ce processus implique plusieurs étapes : parsing, mapping, tri logique et moyennage.