Interprétation interactive de documents structurés : application à la rétroconversion de plans d'architecture manuscrits

Achraf Ghorbel

Thèse

THÈSE INSA Rennes
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le grade de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Achraf Ghorbel
ÉCOLE DOCTORALE : MATISSE
LABORATOIRE : IRISA – UMR6074

Interprétation interactive de documents structurés : application à la rétroconversion de plans d'architecture manuscrits

Thèse soutenue le 11 Décembre 2012
devant le jury composé de :

M. Christian Viard-Gaudin
Professeur à l'Université de Nantes / *Président*

M. Thierry Paquet
Professeur à l'Université de Rouen / *Rapporteur*

M. Jean-Yves Ramel
Professeur à l'Université de Tours / *Rapporteur*

M. Abdelmajid Ben Hamadou
Professeur à l'Université de Sfax / *Examinateur*

M. Eric Jamet
Professeur à l'Université Rennes 2 / *Examinateur*

M. Eric Anquetil
Professeur à l'INSA de Rennes / *Directeur de thèse*

Mme. Aurélie Lemaitre
Maître de conférences à l'Univ Rennes 2 / *Co-encadrante de thèse*

Remerciements

Table des matières

Introduction

Cadre de la thèse : projet ANR

Cette thèse entre dans le cadre du projet ANR-Mobisketch. Ce projet est un travail collaboratif entre :

- l'équipe *IntuiDoc* de l'*IRISA* pour les recherches autour de l'interaction homme-document ;
- l'équipe *LPE* du laboratoire *CRPCC* pour les aspects cognitifs et les cas d'usages, en partenariat avec la plateforme *Loustic* ;
- la société *Script&Go* pour l'élaboration des démonstrateurs et le transfert industriel du projet.

L'objectif du projet Mobisketch est d'aboutir à un continuum le plus fluide possible entre l'analyse automatique de documents numérisés, et la saisie ou l'édition interactive. Le principe est de reconnaitre un document existant numérisé ou vectoriel pour en extraire une représentation numérique interprétée et pouvoir ensuite l'éditer, en contexte de mobilité terrain, à travers une modalité d'interaction «homme-document» orientée stylo. La figure 1 présente les différentes étapes de ce continuum.

Le projet ANR-MobiSketch suit une méthode de conception centrée utilisateur et s'appuie donc sur un travail d'ergonomie de conception. Ainsi, ce sont les besoins des utilisateurs et leurs caractéristiques qui guident le développement du système et non pas seulement les possibilités techniques. Cette démarche implique de faire participer activement les utilisateurs finaux au processus afin d'appréhender toutes les problématiques liées à leurs futures interactions avec le système. Cette approche centrée utilisateur vise ainsi à fournir des recommandations permettant d'améliorer le système en termes d'acceptabilité, d'apprenabilité ou encore d'utilisabilité.

Le cadre applicatif de *Mobisketch*, est l'analyse de croquis de plans d'architecture.

Mise en place du continuum

Le continuum envisagé nécessite deux phases d'interprétations, et donc deux moteurs d'analyse :

- *l'interprétation a posteriori*, dite également «*rétroconversion*». Cette phase a pour objectif de bien reconnaitre un document structuré après sa composition, par exemple sur le papier ;
- *l'interprétation à la volée*. Cette phase permet l'interprétation du document au fur et à mesure de sa composition, par exemple lors de la l'élaboration du document sur tablette.

Nous considérons qu'avoir des caractéristiques similaires entre les deux moteurs d'analyse est un facteur important pour le bon fonctionnement du continuum.

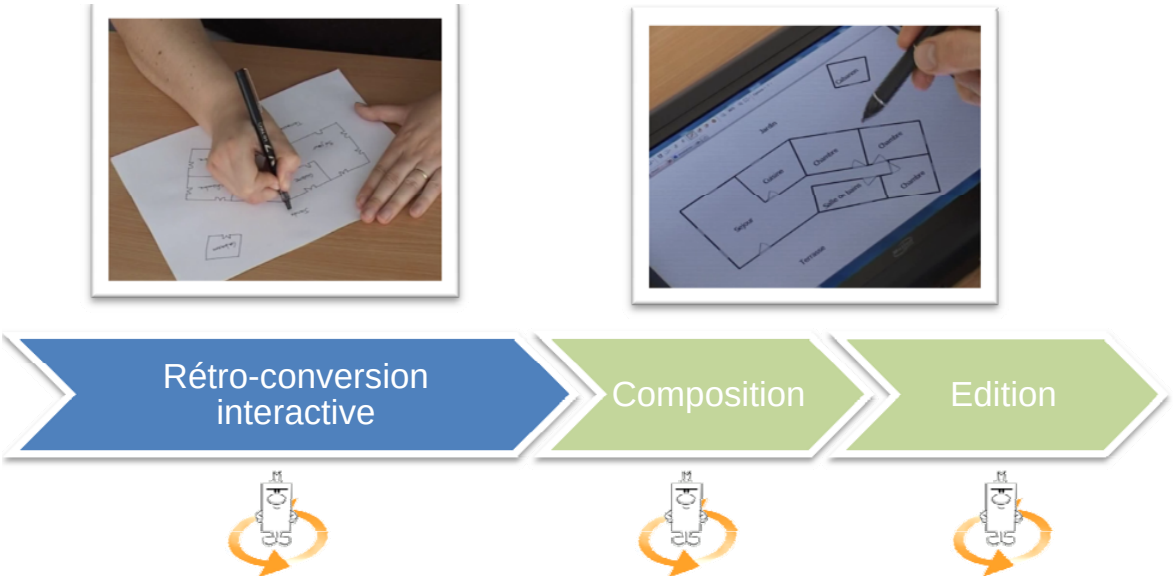


Figure 1 – Schéma global du continuum visé par le projet Mobisketch

Dans cette thèse, nous nous intéressons principalement à la phase d'*interprétation a posteriori*, nécessaire lors de l'étape de rétroconversion interactive. La reconnaissance a posteriori de documents structurés a pour but de réaliser un passage automatique de l'information contenue dans un document existant numérisé ou vectoriel pour en extraire une représentation numérique interprétée et pouvoir ensuite l'éditer. Le support numérique permet de manipuler cette information de manière syntaxique et sémantique. Ce passage automatique permet d'éviter une saisie manuelle longue, fastidieuse et coûteuse.

La fiabilité des résultats est la caractéristique la plus importante pour la l'acceptabilité de ce passage automatique. En effet, si la vérification manuelle a posteriori de

l'ensemble de ce qui est produit par le système est fastidieuse, le gain de temps peut être fortement diminué, voire nul par rapport à une composition numérique complète.

Notre objectif, lors de cette thèse est de concevoir un système interactif de rétroconversion. Cette rétroconversion de documents structurés dégage un ensemble de problématiques, notamment en ce qui concerne les croquis des plans d'architecture :

- la reconnaissance de documents structurés ne consiste pas uniquement à reconnaitre un symbole sous sa forme graphique, comme dans le cas des symboles isolés, mais aussi à tenir compte de son contexte structurel. Ce couplage graphique/contexte structurel introduit une complexité spécifique pour la bonne interprétation de document.
- la reconnaissance de documents structurés nécessite la gestion des données de nature imprécise, due essentiellement à la nature manuscrite des documents : la forme des symboles dessinés est imprécise, et les symboles ne sont pas parfaitement positionnés ;
- l'interactivité exige un temps d'analyse acceptable pour l'utilisateur. Une durée trop longue d'interprétation est ennuyante pour l'utilisateur alors qu'une interprétation trop rapide ne permet pas à l'utilisateur de bien suivre l'évolution du système de reconnaissance et donc de bien interagir avec le système ;
- la présence de l'interactivité dans un processus d'interprétation a posteriori nécessite la présence de plusieurs hypothèses concurrentes pour l'interprétation d'un élément en cas d'ambigüité : c'est alors à l'utilisateur de valider la bonne hypothèse. Ceci peut engendrer de la combinatoire. Une maitrise de la combinatoire est nécessaire pour le bon fonctionnement du système. Il faut d'une part garder les hypothèses concurrentes et d'autre part éviter les hypothèses inutiles.
- le bon compromis interaction/reconnaissance reste un problème à gérer dans un tel système. D'une part l'augmentation de nombre d'interactions améliore le taux de reconnaissance, d'autre part un nombre trop important d'interactions peut devenir fastidieux pour l'utilisateur.
- il n'est pas toujours possible de connaître tous les symboles du document avant de commencer l'analyse ce qui est une difficulté pour la reconnaissance de document. Avoir un mécanisme permettant d'ajouter à la volée de nouvelles classes de symboles facilitera l'auto-évolutivité du système.
- une autre contrainte forte est liée à la cohérence entre le mécanisme de reconnaissance et celui d'édition. Aboutir à un continuum entre un document technique sous sa forme papier et ce même document sous sa forme numérique interprétée nécessite une cohérence et une compatibilité entre le mécanisme de reconnaissance *a posteriori* et le mécanisme d'édition. Pour cela, nous pensons que faire une étude rapide sur les mécanismes de composition et d'édition avant de dé-

velopper la méthode de reconnaissance a posteriori permettra de bien choisir la façon de concevoir notre analyseur.

Nous disposons au départ de ce projet de la méthode DALI [1] [108] pour l'édition et la composition en-ligne des documents structurés. Dans cette thèse, nous allons nous inspirer de cette méthode DALI pour produire notre méthode de rétroconversion, nommée *IMISketch* [2].

DALI est une méthode générique permettant de réaliser des systèmes orientés stylo pour la composition manuscrite de documents structurés en-ligne. Cette approche est basée sur une interprétation *à la volée* des tracés manuscrits. Elle est caractérisée par :

- l'interprétation de formes manuscrites en contexte, pour non seulement reconnaître une forme manuscrite, mais également l'interpréter dans un contexte de document ;
- la généricité : elle est capable d'analyser les différents types de documents structurés (exemples : les plans d'architecture, les schémas électriques, les partitions, etc) ;
- l'exploitation de la connaissance formalisée à l'aide des grammaires GMC-PC, et plus particulièrement des connaissances contextuelles, pour réduire de manière cohérente l'espace des interprétations possibles de la scène considérée. Ceci a pour conséquence de permettre une prise de décision en temps réel, et de limiter les ambiguïtés potentielles ;
- l'intégration de l'utilisateur dans la phase de composition à travers un retour visuel progressif des résultats de l'analyse. En effet, l'analyseur est capable de prendre une décision dès que les informations nécessaires sont disponibles dans la scène. Il peut aussi reporter cette prise de décision s'il manque des informations. L'action de l'utilisateur qui fait suite à ce retour visuel peut alors être utilisée pour valider ou non une prise de décision, ce qui va également permettre de réduire l'espace des interprétations possibles de la scène.

La méthode DALI a été utilisée pour la conception de plusieurs prototypes orientés stylo de composition manuscrite de documents structurés. De plus, elle a fait l'objet d'un transfert industriel qui a mené à la conception du logiciel SCRIPT&GO "SCHÉMAS ÉLECTRIQUES" pour la composition de schémas électriques et une autre version pour la composition des plans d'architecture [127].

La méthode DALI sera utilisée pour la partie «édition» du continuum. Pour assurer l'homogénéité du continuum, nous allons également utiliser une modélisation à base des grammaires de multi-ensembles à contraintes pilotées par le contexte (GMC-PC). Ce

1. **D**éveloppement d'**A**pplication en **LI**gne
2. **I**nteractive **M**ethod for **I**nterpretation of **Sketch**

formalisme est conçu initialement pour la composition et nous allons l'étendre pour l'appliquer au cas de la rétroconversion.

La méthode IMISketch adopte une stratégie basée sur une description grammaticale pour l'interprétation de documents structurés, pilotée par des connaissances *a priori* liée à un domaine spécifique (par exemple : les plans d'architecture). Cette information structurelle est introduite à partir d'un langage visuel permettant l'implémentation d'un analyseur *interactif* et basé sur *une exploration en largeur*. Cette formalisation se base sur une amélioration et une adaptation des grammaires de multi-ensembles à contraintes pilotées par le contexte (GMC-PC), conçues initialement pour l'interprétation des document en-ligne à la volée, pour la reconnaissance a posteriori (*hors-ligne* et *en-ligne*) des documents. Une contribution importante de ce travail consiste à associer aux connaissances structurelles un mécanisme de pilotage du type d'exploration des arbres d'analyse dans le processus d'interprétation.

Originalité de la thèse

L'originalité de cette thèse est d'intégrer explicitement l'utilisateur dans un processus d'analyse *a posteriori*, afin de solliciter l'utilisateur en cas d'ambiguïté. Un cas d'ambiguïté est défini comme un élément du document qui peut être interprété de plusieurs manières. Le principe traditionnellement utilisé est d'attendre que le système de reconnaissance ait terminé l'interprétation du document pour que l'utilisateur vérifie la validité de l'interprétation de l'ensemble du document. Au contraire, dans notre approche, nous intégrons l'utilisateur dans le processus d'interprétation au fur et à mesure de la reconnaissance du document : on parle d'interprétation interactive. Ceci se traduit concrètement par la présentation à l'utilisateur, en cas d'ambiguïté, des différentes hypothèses possibles pour l'interprétation d'un élément. L'utilisateur indique alors l'interprétation correcte.

Nous nous sommes également attachés à gérer la nature manuscrite, et donc imprécise des données manipulées, par une reconnaissance hybride structurelle et statistique des formes considérées.

L'interprétation d'une primitive dans les documents structurés prend en compte ses éléments voisins. Dans le cas des documents bidimensionnels, l'interprétation peut engendrer de la combinatoire. Par conséquent, *IMISketch* définit *un contexte local de recherche*. L'idée est similaire à l'analyseur $LL(k)$ avec lequel le jeton k détermine le contexte dans lequel l'analyseur peut choisir la règle de production à appliquer sans ambiguïté. De la même manière, dans notre analyse bidimensionnelle, nous devons limiter la valeur de k, c'est-à-dire la profondeur de l'analyse d'exploration. Par conséquent,

contrairement à l'analyseur $LL(k)$, l'exploration utilisant des jetons dans un document bidimensionnel ne permet pas de prendre une décision unique concernant la règle de production à appliquer, car k a été choisi volontairement a priori. En outre, parfois, la grammaire n'est pas $LL(k)$: l'analyseur engendre de l'ambiguïté. Pour ces deux raisons, le processus d'analyse n'est pas toujours capable de prendre la bonne décision tout seul, et peut hésiter entre plusieurs hypothèses.

Afin de valider la bonne interprétation, nous allons proposer un processus d'analyse capable, à travers son processus de décision, soit de prendre la bonne interprétation, soit de solliciter l'utilisateur en cas d'ambiguïté.

Cet analyseur est basé sur les caractéristiques suivantes :

- les connaissances structurelles a priori du document sont exprimées à travers un langage visuel basé sur l'écriture de règles de production ;
- l'analyse repose principalement sur une exploration en largeur ;
- l'incertitude est formalisée par l'attribution d'un score à chaque hypothèse sous tendue par une branche de l'arbre d'analyse ;
- l'analyseur travaille sur un contexte local de recherche pour limiter la combinatoire ;
- si les ambiguïtés ne peuvent pas être résolues dans ce contexte local d'une manière automatique, l'utilisateur sera sollicité par l'analyseur pour lever l'ambiguïté.

Ces caractéristiques ont été définies afin d'assurer la meilleure interactivité entre le processus d'analyse et l'utilisateur. Cette interactivité permet notamment de réduire la phase de vérification a posteriori qui peut devenir fastidieuse sur des documents complexes. En effet, en sollicitant à bon escient l'utilisateur sur des phases critiques de l'analyse du document, nous évitons une propagation en cascade d'erreurs de reconnaissance, qui sont très lourdes à corriger a posteriori

Notre nouvelle approche a pour intérêt de surpasser les difficultés des approches «batch» actuelles en évitant une propagation des erreurs d'interprétation au fur et à mesure de l'analyse.

L'ensemble de ces concepts sont intégrés au sein de notre méthode IMISketch (acronyme pour *Interactive Method for Interpretation of Sketch*). Cette méthode est générique et capable d'interpréter *a posteriori* le signal hors-ligne ainsi que le signal en-ligne de documents structurés. IMISketch est mise en œuvre, lors de cette thèse, sur les croquis de plans d'architecture. Soulignons enfin, que l'interactivité de l'approche IMISketch ouvre la possibilité d'avoir une reconnaissance incrémentale, grâce aux informations fournies par l'utilisateur, capable d'ajouter dynamiquement de nouvelles classes de symboles durant le processus d'analyse.

Plan du manuscrit

Le plan de ce manuscrit est le suivant : dans **la première partie**, nous présentons le contexte dans lequel les travaux de cette thèse s'inscrivent et réalisons un état de l'art des méthodes proposées dans la littérature.

Le **premier chapitre** présente les concepts généraux qui s'articulent autour des documents structurés et de leur interprétation. En particulier, nous mettons en avant la complexité de ce processus d'interprétation. Nous passons rapidement sur les méthodes de composition qui peuvent servir pour l'élaboration du continuum et nous détaillons ensuite les principales méthodes de rétroconversion existantes dans la littérature.

Le **second chapitre** présente les différentes étapes du processus d'interprétation a posteriori en mettant en lumière les différentes problématiques de segmentation. Nous nous focalisons sur la catégorisation des approches d'interprétation en détaillant les avantages et les inconvénients de chaque catégorie. Nous présentons également les différentes caractéristiques nécessaires pour réussir l'interactivité dans un système de reconnaissance.

La **seconde partie** de ce manuscrit concerne les spécifications. Cette partie se focalise sur une description de la grammaire GMC-PC existante, utilisée dans notre méthode *IMISketch*.

Le **troisième chapitre** décrit le formalisme grammatical utilisé pour le développement de notre méthode interactive, et pointe les limites de ce formalisme grammatical pour la rétroconversion des documents structurés.

La **troisième partie** de ce manuscrit concerne les contributions de cette thèse, c'est-à-dire la méthode interactive IMISketch.

Dans le **quatrième chapitre**, nous décrivons les différentes parties de la méthode interactive IMISketch. Nous justifions à chaque étape les choix faits pour le développement de notre méthode. *IMISketch* se compose de trois parties principales. La première partie permet la construction des arbres d'analyse en appliquant les règles de production déjà définies. Nous allons présenter dans cette partie quelques stratégies pour la réduction de la combinatoire, telles que le contexte local de recherche qui permet de limiter une zone de document dans laquelle la construction des arbres d'analyse va se dérouler. La construction des arbres est elle-même optimisée. Nous adoptons une exploration hybride qui alterne entre l'exploration en largeur et celle en profondeur. Cette exploration garantit à la fois la présence des différentes hypothèses concurrentes

tout en évitant la construction des nœuds inutiles. La deuxième partie est la phase de reconnaissance a posteriori. Dans cette phase, nous adoptons les grammaires de multi-ensembles à contraintes pilotées par le contexte, conçus initialement pour l'interprétation à la volée des documents. Les améliorations de ce formalisme offrent la possibilité de piloter le type d'exploration des arbres par le formalisme grammatical. Nous gérons également l'interaction avec un classifieur. La dernière partie est la phase de prise de décision. Cette phase gère la sollicitation de l'utilisateur en cas de besoin afin de valider la bonne interprétation. Le processus de décision valide une ou plusieurs règles de production en même temps.

Dans la **quatrième partie**, nous présentons le prototype, les résultats et les évaluations que nous avons effectuées afin de valider notre nouvelle approche de rétroconversion interactive.

Le **cinquième chapitre** décrit un prototype de notre méthode IMISketch appliqué aux croquis de plans d'architecture en se focalisant sur l'implémentation des particularités liées à IMISketch.

Dans le **sixième chapitre**, nous présentons une évaluation des performances d'un système basé sur la méthode IMISketch. Nous montrons d'une part les apports de différentes optimisations des différentes briques de la méthode IMISketch pour la réduction de la combinatoire et le gain en temps de calcul ainsi obtenu. D'autre part, nous mettons en avant la qualité de cette interprétation d'un point de vue reconnaissance de formes. Enfin, nous évaluons globalement le système produit et validons l'impact de l'utilisateur dans le processus de reconnaissance.

Finalement, la conclusion termine ce manuscrit en présentant les perspectives offertes par ces travaux.

Première partie

Principes généraux sur l'analyse de documents structurés

CHAPITRE 1

Concepts généraux

Dans ce chapitre, nous présentons le contexte général autour duquel cette thèse s'articule à savoir la rétroconversion de documents structurés. La rétroconversion (dite aussi reconnaissance a posteriori selon plusieurs chercheurs) est l'interprétation du document après sa composition. Le but est à la fois de fixer la terminologie et les concepts qui seront utilisés dans la suite de ce manuscrit et de mettre en avant les problématiques auxquelles nous devons faire face. En particulier, nous mettons en avant notre volonté de concevoir une méthode de rétroconversion de documents structurés permettant la sollicitation de l'utilisateur dans les cas d'ambigüités. Nous présentons d'abord des notions que nous allons utiliser dans la suite de ce manuscrit notamment définir «un document structuré» ainsi que la signification que nous associons à «l'interprétation de documents structurés». Nous nous focalisons par la suite sur la notion de rétroconversion à travers une catégorisation des différentes méthodes offertes à l'utilisateur pour reconnaitre son document structuré. Nous identifions celle qui nous intéresse dans cette thèse à savoir la rétroconversion des documents structurés. Nous mettons en valeur l'originalité de nos travaux en se positionnant par rapport aux méthodes de rétroconversion existantes. Cette originalité consiste à intégrer l'utilisateur durant la phase de rétroconversion. Enfin, nous mettons en lumière à la fin de cette partie les difficultés inhérentes à la rétroconversion des documents structurés.

1.1 Documents structurés

Dans cette section, nous commençons d'abord par définir la notion des documents structurés, ensuite, nous détaillons le type de document que nous allons étudier dans cette thèse.

1.1.1 Notion de document structuré

Un «document structuré» désigne un document ayant une structure prédéfinie bien établie. Il est composé d'un ensemble de symboles. L'agencement spatial et les positionnements relatifs des éléments du document offrent une information complémentaire et essentielle à l'interprétation du document. Cette information est une caractéristique principale qui distingue les symboles d'un document structuré à des symboles isolés. Les différentes techniques de reconnaissance de tel document se basent principalement sur cette information.

Selon Hilaire [79], les éléments structurés sont composés de :

- traits : généralement l'arc de cercle, le cercle complet, et la ligne droite. Ces courbes sont caractérisées par un ensemble d'attributs tel que l'épaisseur, le style, la couleur, etc ;
- texte : manuscrit ou imprimé ;
- photos.

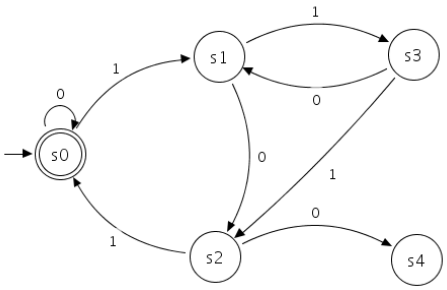
Généralement, les documents structurés prennent l'une des formes suivantes :

- Document imprimé : c'est un document déjà dessiné par un logiciel de dessin technique et imprimé.
- Document vectoriel : c'est document numérique dessiné par des logiciels de dessin vectoriel
- Document manuscrit : c'est un document dessiné à main levée (croquis)

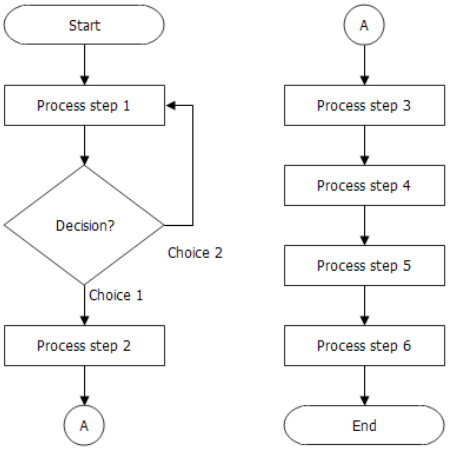
Les problématiques d'interprétation diffèrent d'un type de document à un autre. La qualité de dessin avec un logiciel est nettement meilleure qu'un croquis dessiné à main levée. Dans la littérature, les documents structurés sont de nature très variée. Citons par exemple les diagrammes d'automate à états finis, les partitions de musique, les diagrammes de classe UML, les formules mathématiques, les plans d'architecture, etc. La figure 1.1 illustre des exemples de documents structurés. Dans cette thèse, nous nous intéressons à un type de document structuré bien particulier : les plans d'architecture 2D.

1.1.2 Plans d'architecture 2D

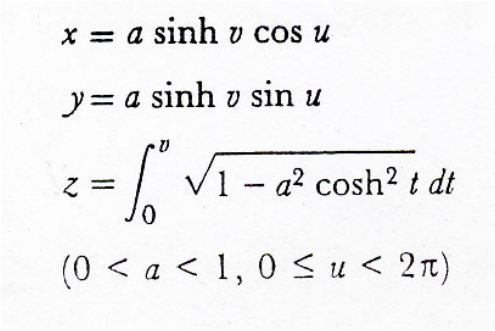
Le plan d'architecture est un document structuré caractérisé par le fait qu'il résulte presque systématiquement de la superposition de plusieurs couches graphiques. Chaque couche est intéressante pour un professionnel particulier, mais pas nécessairement pour



(a) Diagrammes à éléments finis



(b) Flowchart



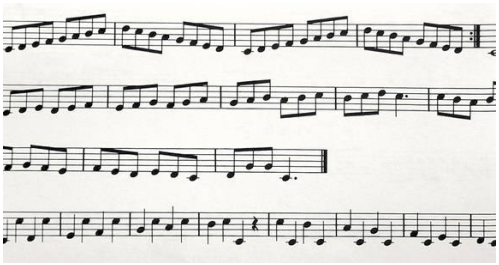
$$x = a \sinh v \cos u$$

$$y = a \sinh v \sin u$$

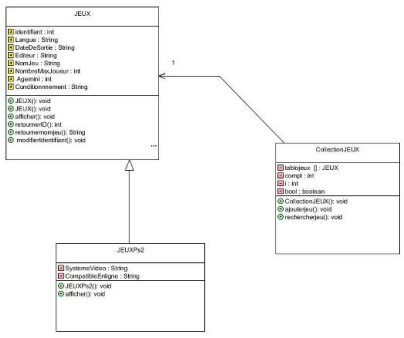
$$z = \int_0^v \sqrt{1 - a^2 \cosh^2 t}\, dt$$

$$(0 < a < 1,\ 0 \le u < 2\pi)$$

(c) Formules mathématique



(d) Partition musicale



(e) Diagramme UML

Figure 1.1 – Exemples de documents structurés

un autre. Nous pouvons résumer un plan d'architecture 2D (figure 1.2 [1]) en trois couches graphiques :

- Une première couche englobe les parties liées aux fondations du bâtiment (maçonnerie). Généralement, cette couche est représentée par un trait épais hachuré.
- Une deuxième couche qui englobe les éléments architecturaux eux-mêmes fenêtres, portes, murs...). Cette couche est souvent représentée par un trait beaucoup plus fin que celui de la premièère couche.
- Une troisième couche comportant des divers symboles ou éléments de construction de dernière œuvre (prise de courant, sanitaires, VMC, carrelage...).

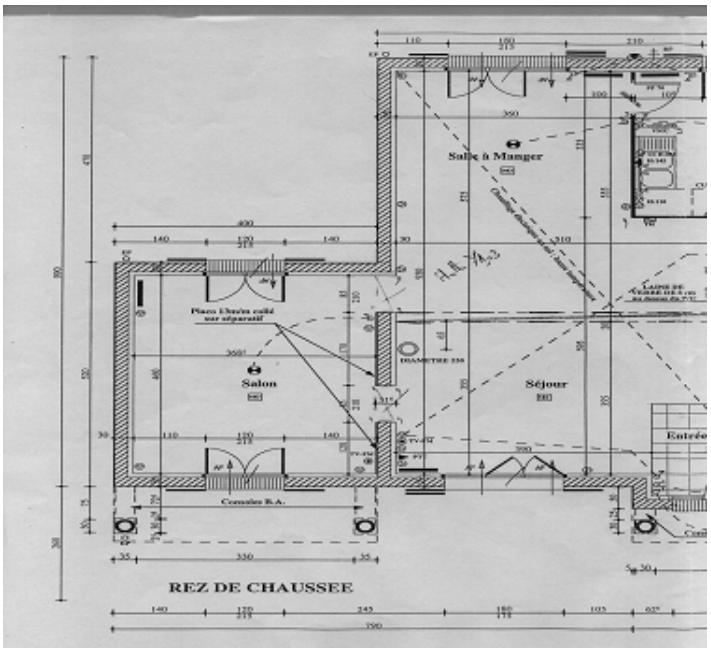
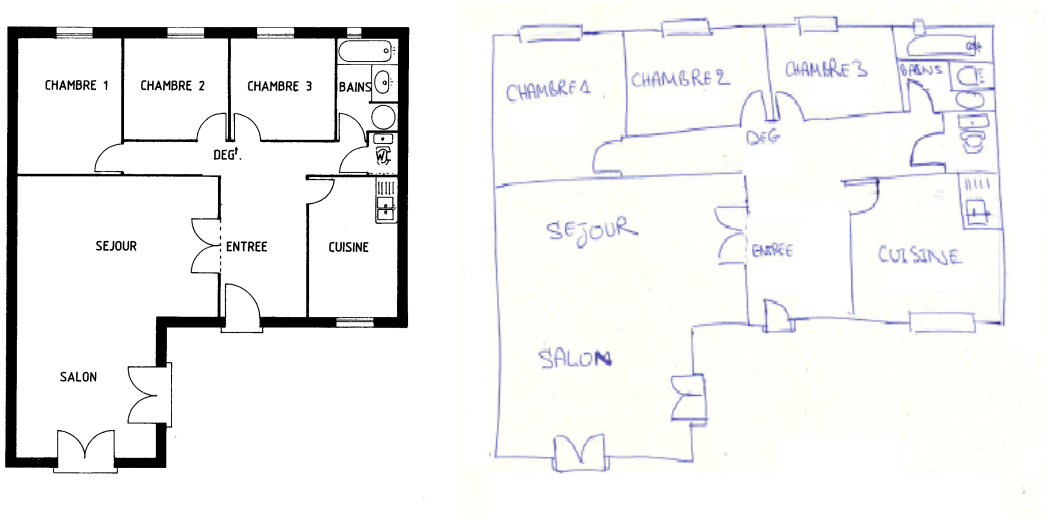


Figure 1.2 – Exemple de plan d'architecte (source FS2i)

Nous nous concentrons dans la suite de cette étude uniquement sur les plans d'architecture destinés aux clients, c'est-à-dire les plans ne contenant qu'un ensemble d'information jugé utile pour le client (figure 1.3) correspondant aux couches 1 et 2. Pour des raison du cadre applicatif du projet ANR, nous commençons par interpréter les plans d'architecture dessinés à main levée sur une feuille de papier. Ces croquis sont généralement des ébauches de conception dessinés par l'utilisateur pour des opérations de métrage, relevés et aménagement de l'intérieur, etc. La figure 1.3(b) illustre un exemple des plans que nous allons interpréter dans cette thèse.

1. FS2i est une PME spécialisée dans le logiciel pour le bâtiment

Ces plans contiennent essentiellement des murs, des ouvrants tel que les portes, les fenêtres et les fenêtres coulissantes et des mobiliers (par exemple : les canapés, les lits et les lavabos).



(a) (b)

Figure 1.3 – Exemples de plans d'architecture simplifiés

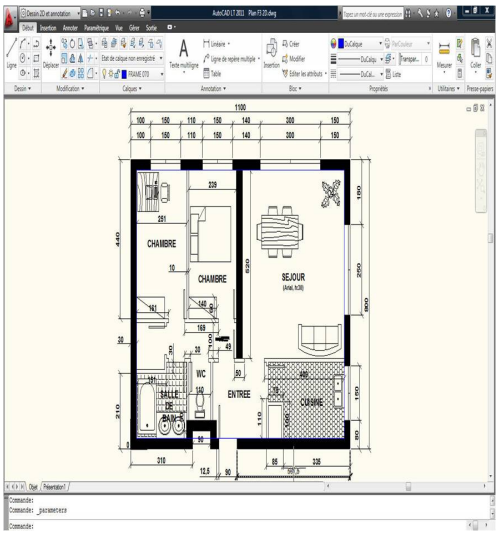
1.2 Composition de documents structurés

La composition de document consiste à construire un document structuré et d'aller vers le monde numérique d'une façon cohérente.

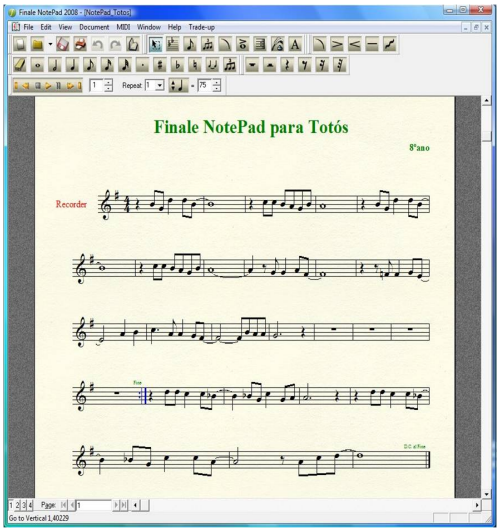
Plusieurs options sont offertes à l'utilisateur pour composer son document technique. La première solution proposée pour dessiner un document est d'utiliser des logiciels basés sur une interaction souris via une interface «WIMP» [2]. Ces logiciels classiques (traditionnels) sont basés sur une interaction graphique orientée souris et boutons. La figure 1.4 illustre quelques exemples de logiciels utilisés pour la composition des documents structurés. Généralement, l'utilisateur sélectionne les symboles à insérer dans son document à travers une interface graphique contenant tous les symboles. Les exemples de tel logiciels sont nombreux. Nous citons par exemple MICROSOFT OFFICE VISIO pour la composition des diagrammes divers, AUTOCAD (Figure 1.4(a)) est largement utilisé pour la production des dessins techniques. FINALE (Figure 1.4(b))permet la composition des partitions musicales. Google SketchUp (figure 1.4(c)) est logiciel de composition des plans d'architecture. Bien que ces logiciels

2. Windows, Icôn, Menu, Pointing device

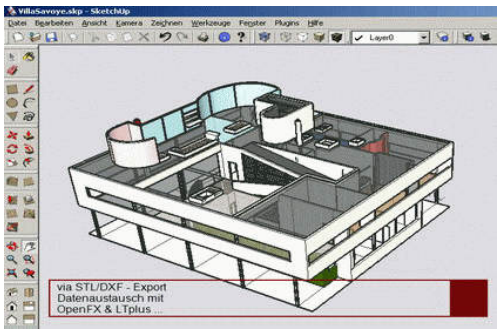
sont généralement très aboutis, ils sont souvent fastidieux à utiliser pour des utilisateurs novices. A la fin de la composition avec ces logiciels, le document sera bien propre et manipulable.



(a) Logiciel *AUTOCAD* pour la composition des documents techniques



(b) Logiciel *FINALE* pour la composition des partitions musicales



(c) Logiciel *Google Sketch Up* pour la composition des plans d'architecture

Figure 1.4 – Exemples des logiciels de composition de documents structurés à base de l'interface WIMP

Une deuxième solution consiste à dessiner un document d'une façon libre, sans imposer des contraintes particulières à l'utilisateur. Cette solution nécessite un système de reconnaissance déclenché après la composition pour bien interpréter le document. Une dernière solution est d'utiliser des logiciels de composition permettant la reconnaissance du document en temps réel. Le système de reconnaissance est intégré dans la phase de composition. Nous intéressons dans la suite de cette partie à ces deux dernières solutions.

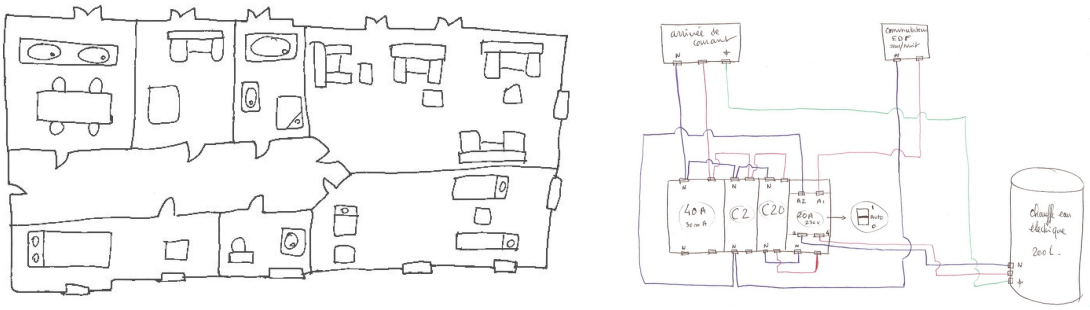
1.2.1 Composition libre de documents structurés

Dans la littérature, il existe plusieurs cas d'usages pour la composition libre des documents structurés ainsi que des techniques assurant la passage du document dessiné sous sa forme numérique interprétée.

1.2.1.1 Usages et propriétés

La composition libre d'un document est la façon la plus rapide et la plus simple pour construire un croquis. Cette possibilité permet à l'utilisateur de composer son document sans des contraintes particulières. Cette composition est effectuée soit sur une feuille papier soit sur une tablette PC via un stylet électronique.

La figure 1.5 illustre quelques exemples de croquis de documents structurés dessinées à main levée sur une feuille de papier.



(a) Croquis de plan d'architecture (b) Croquis de schéma électrique

Figure 1.5 – Exemples des croquis dessinés à la main

L'avantage de cette composition est la conservation du processus créatif de l'utilisateur. En fait, la séparation entre la connaissance et le moteur de reconnaissance permet de donner toute la liberté à l'utilisateur pour dessiner son document. L'utilisateur dessine son document comme il le souhaite, sans qu'aucune contrainte ne lui soit imposée. L'utilisateur doit seulement respecter les règles générales concernant un type de document donné. Il peut, par exemple, commencer un symbole sans avoir fini le précédent. De plus, tous les éléments constituant un document structuré sont omniprésents : le contexte structurel pour interpréter un symbole est disponible quand le processus d'analyse entre en jeu. Cette séparation diminue l'importance du temps d'analyse sur la qualité des systèmes de reconnaissance.

Cependant ces système n'offrent pas d'interaction. Ceci favorise la propagation de mauvaises interprétations durant la phase de reconnaissance, car les systèmes de reconnaissance ne détectent pas d'une manière progressive les erreurs d'interprétation.

Nous détaillons dans la section suivante les principales techniques pour ces systèmes de reconnaissance.

1.2.1.2 Système de reconnaissance : rétroconversion

Étant donné un document structuré, la rétroconversion de ce document est le processus qui vise à en trouver une représentation numérique, manipulable par l'ordinateur, la plus proche possible de celle que son concepteur. Contrairement à l'interprétation en-ligne à la volée, la rétroconversion a pour but de reconnaitre un document après sa composition. Le processus d'analyse est généralement déclenché par l'utilisateur. La figure 1.6 [3] montre un exemple d'interprétation a posteriori : le système n'intervient qu'à la fin de la production du document. Le système essaie d'interpréter les tracés et de les transformer en des symboles.

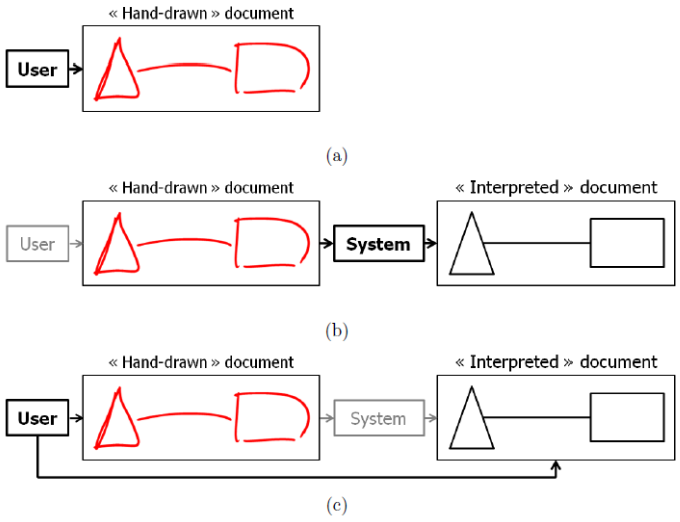


Figure 1.6 – Exemple d'interprétation a posteriori. L'utilisateur commence par dessiner son document (a), puis le système interprète le document (b). L'utilisateur intervient après la reconnaissance pour vérifier et corriger les éventuelles erreurs (c)

Si le document à rétroconvertir est une image, c'est-à-dire un document numérisé (scanné), le signal manipulé est dit hors-ligne, et la rétroconversion est dite «rétroconversion hors-ligne». Ceci signifie que le système a accès à l'information au niveau pixel. Le signal hors-ligne peut représenter à la fois des formes manuscrites et des formes non-manuscrites. La rétroconversion peut également être introduite pour les documents en-ligne, c'est-à-dire des documents constitués d'un ensemble de tracés dessinés par un

3. Cet exemple est pris de la thèse de Macè [108]

stylo électronique. Nous nous focalisons ici sur la rétroconversion *hors-ligne*. La rétroconversion, dans la suite de cette thèse, désigne l'interprétation en-ligne a posteriori et l'interprétation hors-ligne.

La rétroconversion hors-ligne des documents structurés a fait l'objet de plusieurs travaux de recherches. Plusieurs techniques ont été développées pour la reconnaissance de documents structurés spécifiques, c'est-à-dire qu'elles ont été conçues pour un domaine spécifique. La rétroconversion des partitions musicales a fait l'objectif de plusieurs travaux de recherche [92] [112] [133]. Contrairement à ces méthodes, notre but est de développer une méthode *générique* capable d'interpréter les documents de différents domaines, même si notre méthode sera appliquée initialement sur les plans d'architectures.

Plusieurs travaux ont été développés pour la rétroconversion des plans d'architecture. Shio propose le prototype Sketch plan [147] [12] pour l'interprétation hors-ligne des plans d'architecture manuscrits. Valveny et Marti [162] utilisent la correspondance de modèle déformable pour reconnaître symboles architecturaux manuscrits.

Messmer [116] a proposé une méthode basée sur le réseau de contraintes consistant à rechercher l'isomorphisme dans un sous graphes afin d'extraire les descripteurs d'un symbole. Cette méthode a été utilisé pour les plan d'architecture dessinés à main levée. Dans les approches basées sur le réseau de contraintes, les symboles graphiques sont représentés par un ensemble de contraintes de descripteurs géométriques. Ces contraintes sont propagées à travers l'hiérarchie du réseau. L'un des avantages des réseaux de contraintes est la possibilité d'avoir une méthode incrémentale capable d'ajouter de nouvelles classes de symboles. Malgré les efforts de Ah Soon [5] pour rendre la méthode plus claire et plus directe en utilisant la description des symboles par les contraintes de Pasternak [125], l'implémentation de cette méthode reste très complexe. Dosch [54] s'est basé sur cette technique pour la reconnaissance des plans d'architecture.

Lladós [105] propose une méthode structurelle pour la reconnaissance des symboles et des textures. Cette méthode a été appliquée à des plans d'architecture imprimés ou dessinés à la main. Elle est basée sur les techniques des correspondances de chaînes. Lladós considère les symboles et les textures comme un ensemble de régions, c'est-à-dire des formes convexes dans le graphique, avec un agencement particulier.

Ahmed [7] propose également une méthode pour la rétroconversion des plans d'architecture basée sur trois étapes principales : la première phase est la phase de segmentation. Cette étape permet de détecter les murs externes (les murs dessinés par des traits épais) et séparer le texte des graphiques. La deuxième est une phase d'analyse structurelle. Elle consiste à détecter les murs dessinés par des traits fins. La troisième phase est la phase sémantique permettant la détection des portes, fenêtres et les chambres

en utilisant les descripteurs SURF [7] [18]. Dosh [54] a utilisé ces mêmes étapes pour la reconstruction des plans d'architecture en 3D. Nous pouvons citer également les méthodes de reconnaissances de schéma [29] [125] [75] [93] [14]. Toutes ces méthodes sont spécifiques et ne sont pas interactive. L'utilisateur a besoin d'un passage après la phase de rétroconversion pour corriger les éventuelles erreurs. De plus, les méthodes proposées ne peuvent pas prendre en compte des nouveaux symboles durant l'analyse. L'utilisateur est obligé de définir tous les symboles pouvant exister dans le document lors de la conception du système.

Les méthodes de rétroconversion citées précédemment sont destinées à un domaine bien spécifique. Il existe également quelques méthodes génériques.

Notowidigdo [123] propose le système générique «UDSI» [4] pour la reconnaissance hors-ligne. Ce système permet la distinction entre les symboles du document et les caractères textuels.

Messmer [116] a proposé une méthode générique de reconnaissance incrémentale pour la reconnaissance des diagrammes. Cette méthode est évolutive et offre la possibilité d'ajouter des nouveaux symboles lors de l'analyse. La limite majeure de cette méthode est la combinatoire engendrée. L'interactivité pour ce système n'est pas conviviale vu le temps de calcul de la rétroconversion. D'autres techniques, notamment DMOS [48], se basent sur le filtre de Kalman [104] qui est un processus de prédiction-vérification permet de rechercher la position de la suite d'un segment en fonction de la partie du segment déjà détectée.

Comme pour le signal *hors-ligne*, il existe des méthodes *en-ligne* destinées à un domaine spécifique. Nous citons par exemple [34, 60, 65, 96, 155, 172, 16] pour la rétroconversion en-ligne de formules mathématiques. Plusieurs méthodes ont été développées pour la rétroconversion en-ligne des digrammes UML [100, 175] et les circuits électriques [59].

D'autres méthodes sont génériques et capable d'interpréter les documents de différents domaines telles que SketchRead [9] FREEFORM [128].

Les rétroconversions hors ligne et en ligne a posteriori partagent les mêmes problématiques, puisque nous allons chercher à indexer et reconnaître des documents existants. La différence entre l'analyse hors ligne et l'analyse en ligne a posteriori est caractérisée surtout au niveau :

– du signal traité : l'entrée à traiter qui n'est plus un signal associé à l'encre électronique (cas d'analyse a posteriori), mais une image ;

4. User-Directed Sketch interpretation

– du bruit : la quantité de bruit dans les documents scannés est très importante par rapport aux documents électroniques.

En résumé, nous pouvons dire que théoriquement les techniques adoptées pour l'interprétation des documents structurés hors ligne sont exploitables pour la rétroconversion en ligne a posteriori, mais pas inversement (c'est-à-dire que les techniques d'interprétation en ligne a posteriori ne sont pas directement utilisables pour des interprétations hors ligne).

Malgré les recherches actives dans ce domaine, les systèmes de rétroconversion sont encore trop limités. Les méthodes de rétroconversion [47] [58] [109] sont peu performantes surtout dans le cas des documents manuscrits où nous devons gérer l'imprécision. Un conséquence directe est qu'il est alors nécessaire de procéder à une une phase de vérification/correction. Cette phase peut s'avérer très fastidieuse, d'autant plus si les erreurs d'interprétations sont nombreuses. Cette limite est principalement dûe aux phénomènes de propagation d'erreurs : la mauvaise interprétation d'un tracé a un impact sur la suite de l'analyse et peut parfois entrainer des mauvaises interprétations en cascade. L'utilisateur doit alors parcourir tout le document pour corriger les éventuelles erreurs.

De plus, la combinatoire associée à la rétroconversion peut se révéler très grande en raison de la complexité de l'espace des interprétations possibles du document. Vu l'absence d'un système idéale pour la reconnaissance a posteriori la rétroconversion reste aujourd'hui encore un challenge scientifique.

1.2.2 Composition de documents structurés numériques par un logiciel de reconnaissance en temps réel orientée stylo

Comme la composition libre, la composition par des logiciels de reconnaissance en temps réel orientée stylo possède des caractéristiques. Plusieurs approches existent dans littérature.

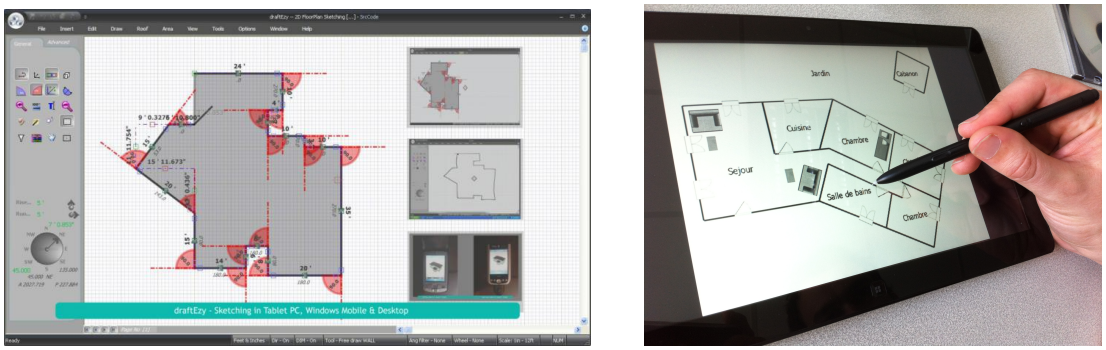
1.2.2.1 Usages et propriétés

On peut directement composer son document à main levée, via un stylet ou une surface tactile ou sensitive telle qu'une tablette numérique. La composition d'un document structuré à travers une interaction orientée stylo nécessite une interprétation des tracés manuscrits dessinés. Le signal obtenu par une telle interaction est dit en ligne.

Les logiciels de composition de documents contiennent un ensemble de fonctionnalités spécifiques à chaque type de document. Parmi les fonctionnalités qu'on peut trouver

dans les logiciels de composition de plan d'architecture, nous pouvons citer le rendu automatique des tracés en lignes droite et l'alignement automatique des lignes (exemple : ligne verticale, ligne horizontale). Nous pouvons trouver également des fonctionnalités permettant la modification des cotations par écriture et la segmentation automatique des tracés complexes.

Cet ensemble de fonctionnalités influe sur la performance du logiciel. L'utilisateur choisit le logiciel qui répond le mieux à son besoin. DraftEzy (figure 1.7(a)), Script&Go Plans(figure 1.7(b)) et Nexus sont trois logiciels de composition de documents d'architecture en ligne et à la volée. Nexus est un logiciel commercial de plans relativement simple d'utilisation, mais avec une interaction stylo assez évoluée. DraftEzy offre un ensemble de fonctionnalités supplémentaires par rapport à Nexus. Le processus d'analyse se déclenche à chaque levée de main. Si on dessine un trait (une ligne), le processus convertit ce trait en un mur. Nexus est un logiciel commercial de plans relativement simple à utiliser, mais avec une interaction stylo assez évoluée. DraftEzy est un logiciel très intéressant qui offre des fonctionnalités supplémentaires vis-à-vis de Nexus comme l'ajout d'élément (exemple : porte, fenêtre), et la détection automatique des lignes courbes dans le document.



(a) Logiciel *DraftEzy* (b) Logiciel *Script&Go*

Figure 1.7 – Exemples des logiciels de composition de documents structurés par une interaction orientée stylo

L'interaction «homme-document» permet de détecter progressivement les erreurs d'interprétation et évite la propagation des erreurs durant l'analyse. Malheureusement, cette interactivité casse le processus créatif de l'utilisateur. L'utilisateur doit attendre le résultat de l'interprétation avant de continuer sa composition. Le temps de réponse système est un facteur important pour l'acceptabilité et la rentabilité des système d'interprétation à la volée.

Même si nos travaux portent sur la reconnaissance hors-ligne de document, il est intéressant de faire un point sur l'état de l'art des approches d'interprétation à la volée en-ligne, d'une part parce qu'elles peuvent être source d'inspiration, et d'autre part pour converger vers la conception d'une approche cohérente qui facilitera l'objectif global du projet ANR qui cherche à coupler rétroconversion et composition pour obtenir le continuum papier/document numérique. Nous étudierons plus particulièrement les méthodes d'interprétation à la volée en-ligne, caractérisées principalement par la généricité et l'interactivité.

Une interprétation à la volée en-ligne consiste à analyser les tracés manuscrits au fur et à mesure qu'ils sont ajoutés au document, il s'agit donc d'une interprétation incrémentale des tracés manuscrits. Le terme tracé désigne les coordonnées des points, ordonnées dans le temps, représentant le parcours du stylo entre un poser et un lever. Chaque tracé est isolé et directement accessible ce qui facilite son interprétation. Les méthodes d'interprétation à la volée sont généralement avec retour visuel, c'est-à-dire à chaque lever de stylo, le système interprète le tracé et affiche le résultat de l'interprétation. L'utilisateur va pouvoir valider implicitement (par exemple en continuant la composition du document) ou la rejeter explicitement (par exemple annulant ou supprimant la dernière interprétation). L'interprétation à la volée avec retour visuel met en place un mécanisme interactif de correction d'erreurs durant la phase de composition du document et donc évite la phase de vérification a posteriori car l'utilisateur sait que tous les tracés dessinés «jusqu'ici» sont bien interprétés.

1.2.2.2 État de l'art des systèmes de reconnaissance avec l'interprétation à la volée

Plusieurs méthodes d'interprétations ont été développées pour interpréter à la volée les documents structurés. Certaines méthodes sont des méthodes génériques c'est-à-dire des méthodes qui ont pour vocation d'être applicables à plusieurs natures de documents structurés. Macé [108] propose la méthode générique *DALI* pour la composition des documents structurés. Cette méthode a été appliqué sur les schémas électroniques et les plans d'architecture. James [99] a proposé le système générique appelé SILK qui permet l'interprétation de reconnaissance à la volée avec retour visuel sous forme textuelle dans une autre fenêtre. D'autres méthodes sont spécifiques et dédiées à un type de document donné. L'interprétation à la volée des partitions musicales a fait l'objet de plusieurs travaux [11] [62] [117] [121]. D'autres méthodes ont été conçues pour l'interprétation à la volée des diagrammes UML [77] [36] [50] [52]. De nombreux prototypes sont liés à l'interprétation à la volée de formules mathématiques [67] [157] [160].

L'avantage majeur de l'interprétation à la volée avec retour visuel est l'existence d'une réelle interaction entre l'utilisateur et le système. Dans un système d'interprétation à la volée, l'utilisateur est acteur du processus d'interprétation. Une fois que le stylo est levé, le système interprète le tracé. En termes de complexité d'interprétation, l'interprétation à la volée présente une diminution importante de la complexité du processus d'analyse car le système interprète un seul tracé dans un document déjà interprété. Le processus d'analyse cherche les différentes hypothèses possibles pour interpréter un tracé dans un document stable, c'est-à-dire bien reconnu. Par conséquent, l'interprétation à la volée à travers son interaction évite de propager de mauvaises interprétations.

La figure 1.8 [5] montre un exemple d'interprétation à la volée : l'utilisateur commence par tracer un tracé manuscrit. En levant le stylet, le système intervient et essaie de reconnaître la forme produite par l'utilisateur dans son contexte et la transforme alors en un triangle. L'utilisateur continue la production de son document et après chaque levée de stylet le système intervient et essaie d'interpréter le tracé produit par l'utilisateur. Dans cet exemple l'utilisateur essaie de tracer un rectangle, et il trace à la fin une ligne qui assure la liaison entre le triangle et le rectangle.

1.2.3 Bilan et conclusion

Dans la littérature, nous distinguons deux grandes familles pour la composition des documents structurés. Dans la première catégorie (section 1.2.1), le moteur de reconnaissance est déclenché par l'utilisateur après la composition du document. La deuxième catégorie (section 1.2.2) intègre le moteur de reconnaissance dans le système de composition. Il existe également des méthodes qui laissent à l'utilisateur de choisir le mode de composition, dite des approches hybrides.

Les approches d'interprétation hybrides sont des approches permettant la reconnaissance de document à la volée et a posteriori, en fonction des préférences de l'utilisateur. Ces approches concernent uniquement les documents en-ligne. Ces méthodes aboutissent à un continuum entre un document technique sous sa forme papier et ce même document sous sa forme numérique interprétée pour pouvoir ensuite l'éditer.

Shilman [143] et Gross [72] proposent des méthodes hybrides pour l'interprétation de document. Kurtoglu et Stahovich [90] ont également proposé une technique d'interprétation de schémas de dispositifs physiques : une partie est interprétée à la volée avec des retours visuels mais l'interprétation globale du schéma est effectuée a posteriori.

5. Cet exemple est pris de la thèse de Macè [108]

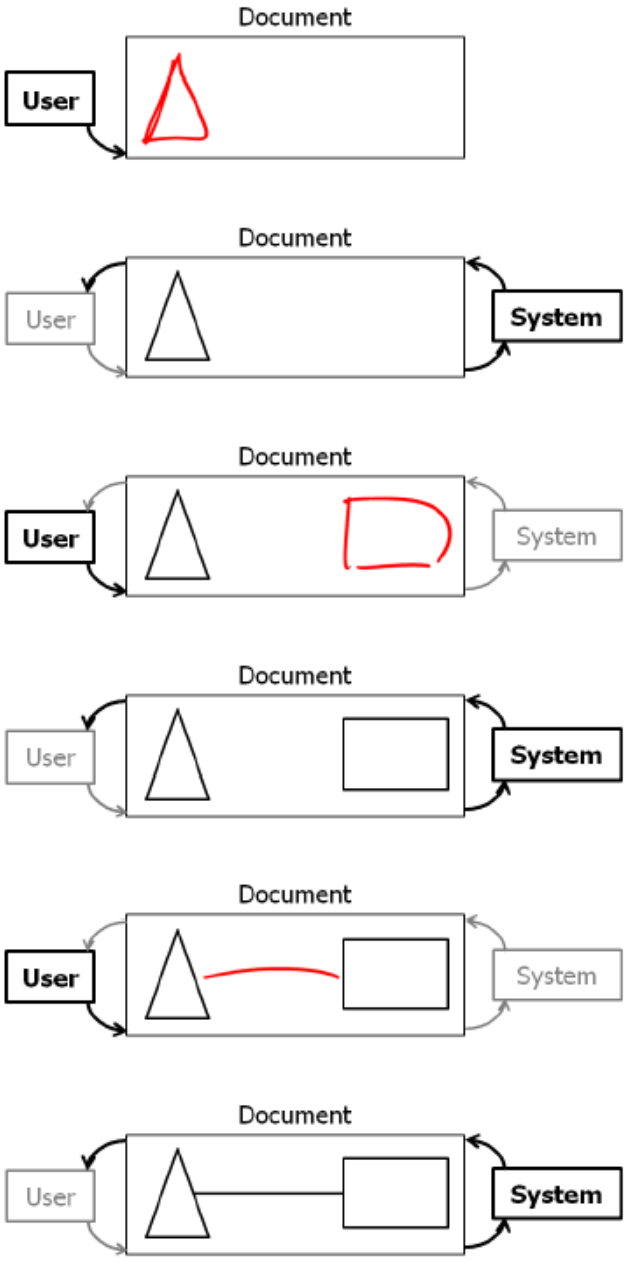


Figure 1.8 – Exemple d'interprétation à la volée

L'idée principale d'un tel système est de reconnaitre un document structuré déjà composé afin de l'éditer et le compléter. Cependant ces systèmes n'offrent pas d'interaction homme-document au sein de leur processus d'analyse. Cela peut avoir comme effet de cumuler les inconvénients des deux natures de processus (à la volée et la rétroconversion en-ligne), à savoir la propagation des erreurs et l'interruption de processus créatif.

En conclusion, chaque catégorie d'interprétation montre des avantages et des inconvénients. Le tableau 1.1 résume les propriétés associées à l'interprétation à la volée et à la le rétroconversion d'un document.

Propriétés	**Interprétation**	
	À la volée	*Rétroconversion*
Interaction Homme-Machine	Oui	Non
Conserver le processus créatif de l'utilisateur	Non	Oui
Détection progressive des erreurs d'interprétation	Oui	Non
Propagation de mauvaises interprétations	Non	Oui
Tolérance au niveau du temps de la prise de décision	Non	Oui

Tableau 1.1 – Comparaison entre l'interprétation à la volée et la rétroconversion

Dans la suite de cette thèse, nous nous intéressons à la rétroconversion des documents structurés. La méthode de rétroconversion que nous proposons dans ce manuscrit est capable d'interpréter à la fois les documents structurés hors-ligne et en-ligne. De plus, contrairement aux méthodes existantes, notre méthode assure une interactivité homme-document afin de bénéficier des avantages de l'interactivité : limitation de la phase de vérification a posteriori et réduction de la combinatoire. Dans la le chapitre suivant, nous focalisons sur les différentes approches existantes pour la rétroconversion de documents structurés.

CHAPITRE 2

Processus et principe de la rétroconversion

Le processus de rétroconversion a pour but d'interpréter un document structuré après sa composition. Généralement, les documents structurés contiennent du texte et du graphique. Nous nous intéressons dans cette thèse uniquement à la partie graphique.

La séparation texte-graphique d'un document scanné est un problème étudié depuis le début des années 80 et a déjà donné lieu à plusieurs publications [1, 61, 106, 159, 164]. Ramel [132] présente une technique de reconnaissance *hors-ligne* de formules chimiques manuscrites imprimées, avec un module de localisation de texte qui extrait des zones de texte pour un système OCR externe.

Le processus général de la rétroconversion des documents structurés est schématisé par la figure 2.1. Plusieurs auteurs [20, 22, 126, 23] ont adopté cette décomposition dans leur travaux de recherche. Le processus de rétroconversion est composé de trois phases : prétraitement, analyse et vérification/correction.

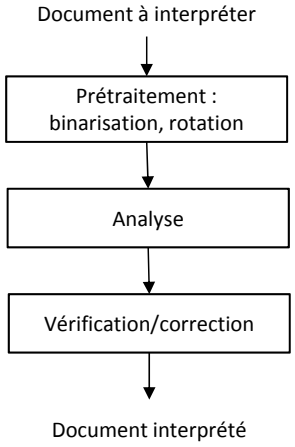


Figure 2.1 – Schéma global pour la rétroconversion des documents structurés

2.1 Prétraitement

La phase de prétraitement consiste à mettre un document dans la meilleure qualité graphique possible. L'acquisition de l'image pour l'interpréter introduit du bruit. Le filtrage du bruit est un traitement supplémentaire pour le traitement hors-ligne. Notowidigdo [123] utilise un filtre gaussien pour absorber le bruit et éviter des erreurs de détection des primitives [131].

Dans l'analyse d'images de documents et la reconnaissance de symboles, la binarisation est toujours une des premières étapes utilisées avant l'étape de reconnaissance. Elle a donc une grande influence sur la performance des étapes suivantes et sur le résultat final. C'est une technique importante dans les applications de traitement d'images.

La binarisation (le seuillage) est la technique de classification la plus simple où les pixels de l'image sont partagés par un seul seuil s en deux classes : noir et blanc. Selon Horaud [81], il existe trois grandes techniques de sélection du seuil s : globale, locale et dynamique.

Dans la méthode de binarisation globale un seuil unique est calculé à partir d'une mesure globale sur toute l'image. La méthode décrite dans [124] essaie de maximiser la variance entre deux classes, tandis que d'autres méthodes dans [88, 130, 115, 39] se basent sur la théorie de maximum d'entropie ou d'entropie floue.

Pour la binarisation locale, la classification d'un pixel dépend non seulement du pixel soi-même mais aussi de ses informations locales. Dans [38], les informations locales sont inclues dans le homogramme qui indique le degré d'homogénéité correspondant à chaque niveau de gris dans l'image. La détermination du seuil se base sur cet homogramme.

Chow et Kaneko proposent une méthode de binarisation dynamique dans [40] où pour chaque point de l'image un seuil dépendant de l'histogramme des niveaux de gris de ses voisins est défini. Pour ne pas calculer un histogramme pour chaque point de l'image, l'image est divisée en blocs d'intersection vide. Pour chaque bloc, l'histogramme des niveaux de gris est calculé et on regarde si cet histogramme est bimodal : si l'écart-type des niveaux de gris du bloc est suffisant, l'histogramme local est alors lissé et approximé par un mélange de deux distributions gaussiennes. Les figures 2.2 et 2.3 illustrent un exemple de binarisation d'une partie du plan d'architecture. L'image binarisée sera l'entrée du moteur d'analyse.

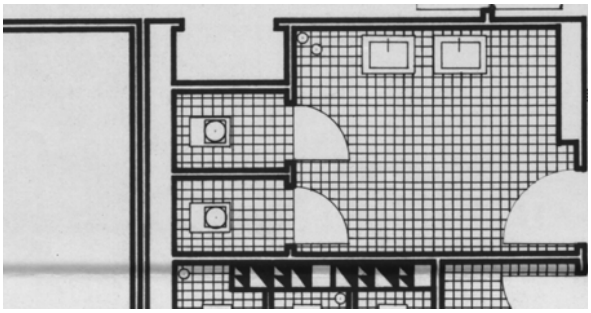


Figure 2.2 – Partie du plan d'architecture

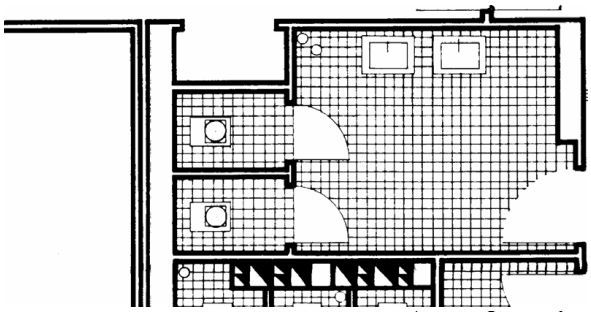


Figure 2.3 – Partie du plan d'architecture binarisée

Le but de notre travail ne porte pas principalement sur cette étape. Nous avons utilisé les méthodes existantes dans léquipe *IntuiDoc*, qui s'appartenant à une binarisation dynamique.

2.2 Analyse

Généralement, la phase d'analyse se compose en deux grandes étapes (figure 2.4) : l'extraction des primitives consiste à récupérer l'ensembles d'éléments de base de l'image binarisée et l'analyse de ces primitives a pour but de bien regrouper les primitives afin de reconnaitre les symboles du document.

2.3 Analyse : extraction des primitives

L'extraction des primitives permet de trouver les éléments de base du document. Les primitives ainsi obtenues seront regroupées pour former les symboles du document. Ces primitives sont un premier niveau de représentation intermédiaire entre l'image binarisée et la représentation objet de l'image qui est le niveau d'abstraction idéal. Étant donné une courbe discrète, le tracé, il s'agit de retrouver les primitives (cercles,

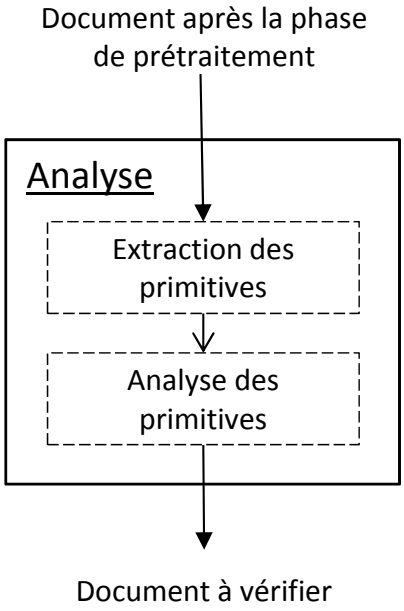


Figure 2.4 – Les étapes de la phase d'analyse

arcs, segments...) qui caractérisent cette courbe. Les primitives sont ensuite structurées dans des objets graphiques de plus haut niveau.

2.3.1 Méthodes existantes

Dans la littérature, il existe plusieurs méthodes pour l'extraction des primitives. L'extraction des primitives hors-ligne se base sur du traitement d'image au niveau pixel pour détecter les primitives graphiques souhaitées. Tombre [158] propose une méthode à base de squelettisation permettant d'analyser le graphe construit à partir du squelette. Les méthodes basées sur la squelettisation sont couramment utilisées. A partir du squelette de l'image, elles permettent d'extraire un graphe de pixels. Le graphe est ensuite analysé pour construire des primitives graphiques (arcs, segments...).

Ablameyko [3] propose une méthode à base de de détection de contours basée sur l'analyse des contours des formes d'un document pour extraire des primitives graphiques.

D'autres méthodes sont basées sur les graphes de composantes. Ces méthodes se distinguent de celles des autres catégories car elles exploitent les composantes connexes et leurs relations topologiques. Il s'agir d'identifier et d'extraire les composantes connexes en tant que primitives graphiques, en segmentant l'image en régions de pixels interconnectés ayant les mêmes caractéristiques. Dans le cadre de cette thèse, nous travaillons à partir d'images qui ne représentent qu'aucun symbole isolé. À première vue ces méthodes ne semblent donc pas très adaptées à notre problème. Cependant les compo-

santes connexes, telles qu'elles sont définies dans [146], peuvent composer le fond autant que la forme d'une image.

Une autre méthode à base de parcours de forme a été proposé par Dori [53]. Cette méthode permet l'extraction de graphes représentant les axes médians des composantes. Une nouvelle technique à base de décomposition en régions [37, 33] segmente les composantes connexes en un ensemble de régions distinctes. Vaxivire [163] propose la décomposition en mailles. Le principe est de sous-échantillonner l'image en un ensemble de mailles. Ces mailles correspondent alors à des sous-ensembles de pixels de l'image. A partir de ces mailles, les primitives graphiques sont extraites. Nous avons également des approches à base de segmentation d'objets : le principe est de segmenter des objets de l'image comme les lignes, les arcs et les ellipses.

Il existe également des méthodes par une approximation polygonale. Ces méthodes recherchent la représentation polygonale globale de la courbe qui vérifie certaines contraintes ou optimise certains critères. Elle consiste à remplacer la courbe initiale par des segments de droite afin d'obtenir un polygone contenant un minimum de points et ajustant au mieux la courbe initiale.

Plusieurs auteurs proposent des méthodes à base d'approximation polygonale. Ces méthodes recherchent de présenter la courbe en un polygone global vérifiant certaines contraintes et optimise certaines critères. La plupart de ces méthodes peuvent être classées en trois catégories. La première catégorie contient les méthodes effectuant un parcours de la courbe originale, un sens de parcours et un point initial sont convenus. Le point initial sera un sommet du polygone. Chaque fois qu'un critère n'est pas vérifié, un nouveau sommet sera positionné. La deuxième catégorie contient les méthodes effectuant un découpage récursif : on prend deux points de la courbe et on subdivise la courbe entre ces deux points jusqu'à ce que l'ensemble des segments satisfasse les critères requis. La troisième catégorie contient les méthodes considérant l'approximation comme un processus d'optimisation globale. Il existe aussi des méthodes basées sur la détection des points de courbure du tracé à segmenter [170].

2.3.2 Choix des primitives

Le choix des primitives est assez variable d'une approche à l'autre. Dans la littérature, il existe des méthodes qui considèrent les segments et les composantes connexes [48] comme des primitives. Hammond [78], Lank [100] et Freeman [64] choisissent les formes géométriques (rectangle, arc, etc.). Ablameyko [2] donne plus de précision sur les primitives et considère que les segments fins, les segments pointillés, les arcs et les zones hachurées sont suffisants pour avoir de l'information nécessaires sur

les images. Zheng [174] définit les primitives sous forme de segments, arcs et cercles. Huang [84] propose les polygones et les cercles pour former l'ensembles de primitives de l'image. Messmer [116] et Notowodgodo se basent sur un ensemble de segments pour la rétroconversion hors-ligne. Shio [147] donne plus de spécificité aux segments en les divisant en segment fins et segments épais. Gennari [66] considère les segments et les arcs comme l'ensemble de primitives nécessaires pour la rétroconversion en-ligne des diagrammes.

Un ensemble de primitives trop haut niveau peut engendrer des remises en cause de ces traitements bas niveau par l'analyseur, ce qui s'avère complexe à maîtriser. La richesse de la représentation a une importance mais elle peut également induire de l'erreur. Un ensemble de primitives trop basique peut engendrer une grande combinatoire au niveau de l'analyseur pour gérer ces hypothèses élémentaires de segmentation.

2.4 Analyse : analyse des primitives

L'étape d'analyse des primitives consiste à collecter et regrouper l'ensemble de primitives graphiques structurelles comme le segment, l'arc, etc, faisant partie d'un symbole, afin de bien le reconnaître. Mais selon le paradoxe de Sayres [138] : «il faut segmenter pour reconnaître, mais il faut reconnaître pour segmenter».

Dans le domaine de composition de document, cette segmentation en primitives graphiques structurelles offre la possibilité de dessiner un symbole de manières différentes (figure 2.5), en particulier avec un nombre différents de tracés et permet aussi de s'abstraire potentiellement de la difficulté liée au fait qu'un tracé peut faire partie de plusieurs symboles différents.

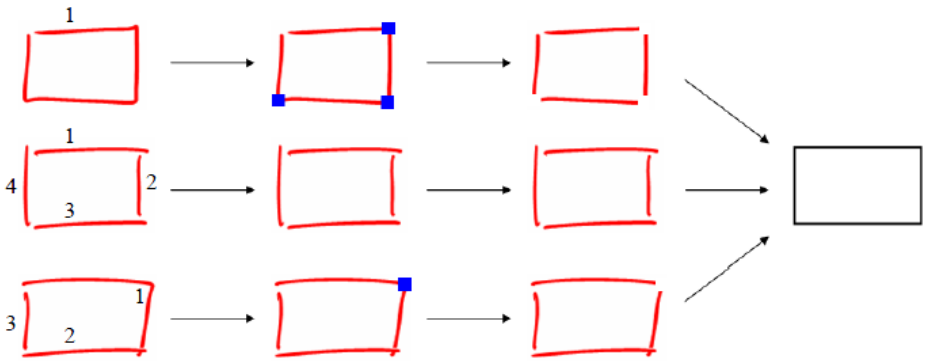


Figure 2.5 – Avantages du processus de segmentation : un rectangle peut être dessiné à l'aide d'un nombre quelconque de tracés

La liste des primitives choisie alimente le moteur d'analyse. Le bon choix de type de primitive évitera les sur-segmentations et les sous-segmentation. Dans la littérature, il existe des analyseurs de rétroconversion qui nécessitent un seul passage pour regrouper les symboles de document et d'autres qui sont basés sur un double passage pour bien reconnaitre les symboles.

Dans cette section nous présentons les approches d'analyse principalement utilisées pour la rétroconversion de documents structurés.

Certains travaux considèrent qu'un seul passage suffira pour le regroupement des primitives. Plusieurs auteurs [74, 139, 144, 145] considèrent un problème d'optimisation et définissent des fonctions de coût C d'un regroupement v_i des primitives graphiques structurelles considérées (ou tracés). Un exemple de fonction est décrit ci-dessous.

$$C(V_i) = \phi(R(V_0), R(V_1), ..., R(V_n)))$$

- R est le coût élémentaire de la meilleure reconnaissance en symbole V_i (une valeur moins élevée modélisant une meilleure confiance en la reconnaissance).
- ϕ fonction de combinaison de coûts élémentaires.

Cette approche exige l'utilisation d'un classifieur de symboles dont on peut obtenir un indice de confiance du symbol identifiée. De plus ces approches utilisent les contraintes contextuelles spatiales ou temporelles (utilisées uniquement dans le cas des interprétations à la volée) pour réduire l'espace de recherche [74]. Rappelons que la rétroconversion des documents structurés ne peut utiliser que les contraintes contextuelles structurelles. Shilman et Viola [144, 145] et Hammond [78] proposent une méthode basée uniquement sur des contraintes structurelles. L'idée de cette méthode est de construire un graphe de proximité, c'est-à-dire chaque primitive est un sommet de graphe et chaque arc relie les primitives éloignées de moins d'un nombre prédéfini de pixels. Uniquement les regroupements des primitives reliés dans le graphe sont considérés.

D'autres travaux présentent des approches de reconnaissance en deux temps analysent les documents en deux passages. Le premier passage consiste à détecter les symboles faciles à reconnaitre et le deuxième passage a pour but de compléter la reconnaissance du document en se basant sur les symboles reconnus lors du premier passage. Le prototype SIMULINK est une méthode proposée par Kara [90, 89] pour le regroupement de primitives. Cette méthode consiste à trouver les symboles faciles à reconnaitre appelés marqueurs. Ces marqueurs sont choisis a priori [89]. Ils seront considérés comme des points d'encrage pour aider la suite de l'analyse. Le regroupement des primitives consiste à exploiter les marqueurs trouvés précédemment pour

regrouper les autres primitives en symboles [89]. Une fois que les primitives sont bien regroupées, le système passe à la reconnaissance.

Par le même principe, Zheng [175] propose une méthode qui consiste à regrouper les symboles en deux temps. L'idée est de commencer par reconnaitre les symboles indépendants du contexte structurel (dit «prégnants»). La deuxième étape consiste à reconnaitre les autres symboles dits «réguliers».

L'inconvénient de ces méthodes est la dépendance par rapport aux éléments reconnus a priori (marqueur ou prégnant). La mauvaise reconnaissance de ces symboles implique une réduction des performances de regroupement du document.

Dans la littérature, trois catégories d'approches sont présentes pour la rétroconversion des documents structurés : les approches basées sur des heuristiques, d'autres basées sur des approches statistiques et celles basées sur une modélisation structurelle explicite. Plusieurs approches intégrent statistiques au sein des approches structurelles.

2.4.1 Approches basées sur des heuristiques

Les heuristiques sont généralement utilisées pour analyser les documents structurés de nature assez simple contenant peu de symboles différents. Les approches à base d'heuristiques sont très dépendantes de l'application. Elles ont notamment été exploitées pour la reconnaissance des diagrammes UML. Lank [100] considère que les diagrammes UML sont constitués de deux natures d'éléments : les symboles de diagrammes et le texte. L'heuristique développée permet la distinction entre ces deux natures. L'auteur se base sur la taille des objets. En fait, il suppose que la taille d'un texte est plus petite que celle d'un symbole UML. TAHUTI [139] se base également sur des heuristiques. Ces approches sont difficilement exploitables dès que le document devient plus complexe. De plus, il faut les redéfinir entièrement par chaque nouveau type de document à traiter

Il existe également des méthodes à base de détection de points critiques. Elles partent sur le fait que dans le cas continu et en absence de bruit, la transition entre la fin d'une primitive et le début d'une autre est toujours marquée par une discontinuité de première ou de deuxième espèce de la fonction courbure ou de sa dérivée première [165]. Dans le cas discret et en présence du bruit, la discontinuité est traduite par la variation de ces fonctions au voisinage de certains points.

Plusieurs chercheurs comme Freeman [63], Sharma [137], Rosenfeld [135] proposent des méthodes reposant sur la détection de points critiques.

Pour Freeman [63] le critère de décision porte sur la moyenne des différences des angles mesurés entre un point donné et son voisinage. La méthode de Sharma [137]

utilise aussi un voisinage de k pixels pour détecter les points dominants mais de manière différente. Rosenfeld [135] introduisent la notion de k-courbure, en utilisant un modèle à deux segments qu'ils ajustent sur la courbe et retiennent les points pour lesquels la valeur du cosinus de l'angle formé par ces segments est supérieure à toutes les autres dans un voisinage de taille k.

2.4.2 Approches basées sur les statistiques

Les approches statistiques se basent sur la détermination d'une signature du symbole à reconnaître. Ces approches introduisent un système de classification qui analyse un vecteur caractéristique déduit du document [136, 21, 56, 4, 151]. Chaque caractéristique est une représentation vectorielle, matricielle ou structurelle du symbole du document. La qualité d'une caractéristique est évaluée par son pouvoir discriminant : la qualité de distinguer une image d'une autre. Les relations entre les caractéristiques ne sont pas prises en compte.

Certaines caractéristiques sont fréquemment utilisées. Citons par exemple le nombre de points d'intersection entre des segments, orientation, nombre de jonctions dans le symbole, surface, périmètre, rapport hauteur/largeur, etc.

Le but de la reconnaissance statistique est de minimiser la distance entre le vecteur des caractéristiques du symbole à reconnaître et celui du symbole candidat. Ces caractéristiques permettent une reconnaissance et un traitement rapide mais elles sont peu discriminantes. Pour cela, certains travaux font recours à des caractéristiques plus évolués telles que :

- les transformées (Fourier [173], Fourier-Mellin [4]),
- la signature de Radon (R-Signature) [150, 152],
- les moments géométriques et autres moments (Hu [83], Zernike [82, 13], Legendre [156]...)

Une fois les caractéristiques extraites, plusieurs classifieurs peuvent être mis en œuvre, la littérature en propose un large choix : bayésien [97], K plus proches voisins (KPPV), Séparateur à Vaste Marge (SVM).

Les modèles de Markov cachés sont couramment utilisés pour la segmentation et la reconnaissance des données séquentielles, tandis que les champs de Markov se sont avérées être le modèle statistique le plus puissant pour la manipulation des données bidimensionnelles. Nicolas [122] a proposé une méthode statistique pour la rétroconversion hors-ligne des documents structurés qui combine ces deux modèles. Cette méthode a été appliquée sur les manuscrits de Gustave Flaubert. Lemaitre [103] et Montreuil [118]

ont proposé des méthodes basées sur les champs conditionnels aléatoires pour la reconnaissance des différentes parties des lettres.

Kosmala [95] utilise les modèles de Markov cachés pour regrouper les primitives des symboles des formules mathématiques manuscrites. Szummer [149] exploite les Conditional Random Fields (CRF) pour regrouper et reconnaître des diagrammes simples. Le principe de ce formalisme est de modéliser non seulement les dépendances entre les données étudiées et leur étiquette, mais également les dépendances entre étiquettes. Ces dépendances sont apprises automatiquement à partir d'exemples. Les données en entrée sont des segments.

Nayef [120] propose une méthode qui vise à détecter les symboles d'un document technique. Cette méthode a été appliquée sur les plans d'architecture. Elle est basée sur des calculs statistiques inspirés de l'algorithme de groupement de Jacobs [86]. Elle consiste principalement à détecter les formes convexes dans l'image qui caractérisent les symboles recherchés.

Sezin [140] a proposé une méthode de rétroconversion de croquis en-ligne basée sur les Réseaux Bayesien Dynamique. Freeman et Plimmer [64] se sont intéressés à la reconnaissance de graphes de complexités variables.

Les expériences ont montré que les approches statistiques sont bien adaptées pour la reconnaissance des documents manuscrits car elles sont capables de gérer la nature imprécise des données.

Malheureusement, les approches statistiques ne permettent pas une modélisation explicite de la structure de document. De plus, elle nécessité d'une large bases d'exemple pour l'apprentissage. Ces inconvénients rendent l'intégration de l'interactivité avec un utilisateur et faire évoluer les classifieurs difficile à l'implémenter.

2.4.3 Approches structurelles

La reconnaissance structurelle consiste généralement à définir chaque symbole pouvant exister dans un langage visuel, en terme de primitives graphiques qui le constituent et de l'agencement structurel de ces primitives. La nature structurée du document indique que ses constituants sont eux-mêmes structurés. Ces approches permettent une description des constituants du document en terme de primitives graphiques. L'idée globale de la reconnaissance structurelle repose sur l'identification d'un ensemble prédéfini de primitives vérifiant un certain agencement graphique. Dans la littérature, nous trouvons deux façons classiques pour décrire les constituants d'un document structuré : les graphes et les grammaires.

2.4.3.1 Graphes

Les graphes sont basés sur les notions de sommets et arcs ; chaque graphe est composé d'un ensemble de sommets et d'arcs. Chaque sommet correspond à une primitive graphique du document et chaque arc décrit la relation structurelle entre deux primitives. Cette modélisation demande la bonne identification des sommets et des arcs ce qui n'est pas toujours facile à obtenir. Hammond [78] utilise les graphes pour la reconnaissance bas-niveau des documents en-ligne.

Les graphes ont été beaucoup utilisés dans différentes communautés, telles que les bases de données, les langages de programmation, la biologie, et bien sûr la reconnaissance de formes. Ils ont été exploités pour l'interprétation de documents structurés *hors-ligne* de différentes natures. Dans notre contexte, le graphe semble être un mode de représentation pertinent. Il permet de modéliser la structure du symbole. La reconnaissance structurelle peut donc être vue comme un problème d'exploitation de graphes.

Le principe d'une production consiste à trouver un sous-graphe correspondant à la partie gauche de la production dans un graphe analysé et à le remplacer par le sous-graphe en partie droite.

Une fois le graphe construit, cette approche peut utiliser une méthode de parcours de graphe afin d'en extraire des informations et apparier les graphes avec des graphes modèles, ou alors une méthode de grammaire de graphe qui consiste à appliquer un ensemble de règles afin de faire converger les graphes vers les graphes modèles [6, 101, 171, 30].

Bunke [29] exploite les grammaires de graphes pour l'interprétation des organigrammes et circuits électriques. Le système de reconnaissance prend en entrée un ensemble de segments qui seront ensuite organisés sous forme de graphes en fonction de leur proximité graphique. Les productions permettent de transformer un graphe en un autre modélisant les symboles et leurs connexions. Fahmy [58] a proposé une méthode à base des grammaires de graphes pour l'interprétation des partitions musicales. Il considère un ensemble de symboles comme les données d'entrée à son système d'analyse. Les productions permettent essentiellement d'associer des attributs à chaqu'un des symboles en entrée.

Les grammaires de graphes offrent un mécanisme très expressif pour la reconnaissance de formes. Cependant, trois inconvénients majeurs leur sont souvent associés. Tout d'abord, le concepteur a du mal à manipuler ces grammaires, surtout lorsque les productions deviennent nombreuses. De plus, elles sont coûteuses à mettre en œuvre de par leur dépendance aux techniques de recherche d'isomorphismes de graphes, pro-

blème lui-même NPcomplet. Enfin, les grammaires de graphes sont assez mal adaptées à la manipulation de bruit et d'incertitude.

Les grammaires d'arbres [27, 71] règlent un certain nombre de ces problèmes en permettant de ne plus manipuler des graphes mais des arbres. L'analyseur prend un arbre en entrée. Chaque arbre est composé d'un ensemble de nœuds représentant les primitives et de branches représentant les relations entre ces primitives. Un nœud peut avoir plusieurs fils ce qui permet aux grammaires d'arbres de représenter plusieurs connexion par rapport à une même primitive, mais limité à une seule connexion entre deux primitives. Cette représentation est relativement limitée, car elle nécessite une connexion entre les primitives et surtout, son expression est relativement lourde et donc difficile à manipuler pour des objets complexes.

Messmer [116] se base sur les graphes pour le développement d'une méthode incrémenale pour la rétroconversion des document. Les symboles graphiques et les dessins sont représentés par des graphes relationnels. Le processus de reconnaissance est formulé en tant que processus de recherche des isomorphismes entre les graphes des symboles et le graphe de dessin. Ce système a été complètement mis en œuvre et testé avec succès sur un certain nombre de documents manuscrits.

2.4.3.2 Grammaires

Dans cette partie, nous nous focalisons plus spécifiquement sur les grammaires modélisant les approches génériques, c'est-à-dire utilisable pour l'analyse des documents structurés de différentes natures. Nous allons détailler cette section puisque notre méthode de rétroconvesion *IMISketch* utilise les grammaires. L'intérêt de cette section n'est pas de réaliser une étude exhaustive des formalismes grammaticaux qui ont été proposés pour la reconnaissance de formes ou l'interprétation de documents structurés, mais de mettre en valeur les approches majeures, ainsi que leurs avantages et inconvénients pour une utilisation dans le cadre qui nous intéresse.

Une grammaire est un formalisme qui permet de définir une syntaxe, et donc un langage formel, c'est-à-dire un ensemble de phrases sur un alphabet donné.

Définition 1 : formellement une grammaire G d'un langage est définie par un quadruplet (V_T,V_N, S, P) tel que :

- V_T est un *alphabet* : ensemble de symboles terminaux ;
- V_N est un ensemble de symboles non terminaux, avec $V_T \bigcap V_N = \emptyset$;
- S est *l'axiome* : le symbole de départ, avec $S \in V_N$;

– P est un ensemble de règles de productions, ou plus simplement appelées *production* (appelées aussi des règles de ré-écriture) ; Une production $p \in P$ a généralement la forme suivante : $\alpha \rightarrow \beta$

. α est la partie gauche de production, avec $\alpha \in (V_N \bigcup V_T)^+$;

. β est la partie droite de production, avec $\beta \in (V_N \bigcup V_T)^*$;

Le processus de remplacement de la partie gauche de la production par la partie droite de la production est appelé *dérivation*.

Définition 2 : une phrase est un séquence $v \in (V_N \bigcup V_T)^*$ de terminaux ou non terminaux. Une phrase terminale est une phrase qui ne contient que des terminaux.

Le formalisme grammatical peut être caractérisé par plusieurs critères. Nous proposons le classement décrit dans le tableau 2.1. Dans ce tableau, nous définissons deux critères pour catégoriser les différents formalismes grammaticaux : la relation entes les symboles.

Relation	**Grammaire**
Opérateur	PDL [31, 142]
	Grammaire de position [42, 46, 44]
	Grammaire de position étendue [43]
	Grammaire de dessin [41]
Ensemble de fonctions	Grammaire relationnelle [49]
	Grammaire d'adjacence [87, 68]
	Grammaire PLG [70]
	EPF [48]
	GMC [111]
	GMC-PC [107]

Tableau 2.1 – Classification des grammaires

Les grammaires peuvent être catégorisées selon les relations utilisées entre les symboles de la production. Ces grammaires assurent cette relations à travers des opérateurs, des points de connections ou des fonctions.

2.4.3.2.a Grammaires à base d'opérateurs

Les opérateurs consistent en un ensemble de mots bien définis pour représenter la relation entre les différents symboles. Les règles de production sont considérées comme une concaténation linéaires des éléments reliés par un opérateur. La règle de production est de la forme $c \rightarrow a + b$ avec c est le symbole formé à partir de a et b et l'opérateur $+$

indique la relation particulière entre a et b. Le langage de description (*Picture Description Languages*) (PDL) [31, 142] décrit les symboles avec les deux points extrémités. Les grammaires PDL sont des variations autour des grammaires mono-dimensionnelles afin de manipuler plus facilement des données bidimensionnelles. PDL est l'une des premières tentatives pour décrire les modèles d'un document en utilisant un langage formel. En PDL, chaque primitive possède deux points de connexion : la tête et la queue. Quatre opérateurs binaires, désignés par +, -, x et *, sont définis pour combiner une paire d'expressions PDL. D'autre part, l'opérateur ˜ est utilisée pour inverser la tête et la queue d'une primitive ou une expression PDL. La figure 2.6 montre la manière dont les primitives sont connectées à l'aide de ces quatre opérateurs binaires.

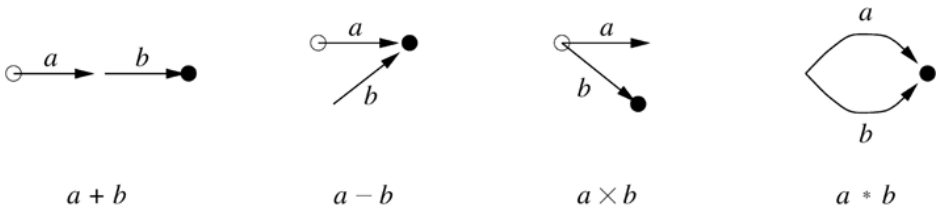


Figure 2.6 – Les quatre opérateurs de PDL

Les grammaires de position (*Positional Grammars*)(PG) [42, 46, 44] expriment les relations en terme de positions et jointure. Les éléments réduits par une production sont définis comme une séquence bidimensionnelle : des opérateurs permettent de définir l'emplacement d'un élément en fonction de ceux précédemment énoncés. Une production est de la forme :

$$A \rightarrow x_1 R_1 x_2 ... R_{m-1} x_m$$

où $A \in V_N, x_i \in (V_N \bigcup V_T)^+$ et R_i est une séquence de position relatives permettant de positionner l'élément x_{i+1} relativement aux éléments $x_1, ..., x_i$. L'analyse de document en utilisant ce formalisme est pilotée par les positions R_i modélisées dans le formalisme : *rechercher un élément positionné à l'emplacement R_i par rapport à des éléments donnés.*

Un inconvénient de ce formalisme est de ne pouvoir modéliser le positionnement d'un élément que relativement aux précédents dans la production. Les auteurs proposent dans [43] une extension permettant notamment d'éliminer cette difficulté. Les grammaires de position étendues (*Extended Positional Grammars*)(XPG) [43] sont utilisées pour la reconnaissance de langages visuels manuscrits dans [45].

Les grammaires de dessin (*Sketch Grammars*)(SG) [41] sont une amélioration des grammaires de position pour la description et l'interprétation des croquis. Ce genre

de grammaire n'utilise pas uniquement les relations de positions mais aussi les relations temporelles. L'inconvénient des grammaires de positions et de ces dérivés est l'insensibilité au contexte et surtout nécessite de connaître les règles temporelles, ce qui permet de reconnaître les éléments d'un document mais pas d'en établir la structure ; en particulier, il est difficile voire impossible de modéliser qu'une même forme peut être interprétée d'une façon différente dans deux contextes différents.

Ces grammaires sont simple à utiliser. Malheureusement les grammaires basées uniquement sur les opérateurs montrent des limites d'expressivité.

2.4.3.2.b Grammaires à base de fonctions

Les grammaires de cette catégorie utilisent les fonctions pour décrire les relations entre les différents symboles. Chaque règle de production est définie par un ensemble de fonctions représentant la relation entre les éléments. Soit A une règle de production basée sur les fonctions. Elle peut être présentée comme suit :

$$c \rightarrow \{ab\} incident(a, b)$$

avec *incident* une fonction permettant l'évaluation de la manière dont l'extrémité de a touche le b.

Dans la littératures, il existe plusieurs approches appartenant à cette catégorie. Certaines approches sont dépendantes du contexte et d'autres non. Par exemple, les grammaires relationnelles (*Relation Grammars*)(RG) [49] ne sont pas dépendantes du contexte, tandis que les grammaires de multi-ensembles à contraintes (Constraint Multiset Grammars)(GMC) [111] et les grammaires de multi-ensembles à contraintes piloté par le contexte (Constraint Multiset Grammars-Context Driven)(GMC-PC) [107] définissent explicitement le contexte dans les règles de production. Par ailleurs, dans les grammaires PLG (*Picture Layout Grammars*) [70] et les grammaires d'adjacence (*Adjacency Grammars*) [87, 68], le contexte est exprimé comme une série de symboles dans la partie droite de la production. Mas [114] utilise les grammaires d'adjacence pour l'interprétation en-ligne des croquis. Cette méthode est appliquée sur des plans d'architecture. Elle permet également de bien tenir compte des problèmes d'incertitude en attribuant un degré de tolérance d'erreur à chaque hypothèse possible pour l'interprétation d'un élément. La grammaire EPF (*Enhanced Positionnal Formalism*) [48] définit des opérateurs de positions permettant de qualifier dans quelle position l'analyseur peut trouver le prochain élément. L'inclusion d'un opérateur de factorisation permet à ces formalismes de décrire une relation entre un symbole et deux ou plusieurs symboles en même temps.

L'analyseur d'EPF utilise les opérateurs de position comme l'opérateur *AT(pos)* pour définir les relations spatiales entre les éléments c'est-à-dire trouver la direction de recherche de l'élément suivant. L'analyseur DMOS associé à cette grammaire assure une grande flexibilité pour la définition d'opérateurs de position. L'EPF possède aussi des opérateurs de contexte, appelés opérateurs de sauvegarde, qui permettent aux règles de la grammaire de faire référence à des éléments pour ensuite les chercher lors de la mise en correspondance. Ceci permet d'établir un lien bidirectionnel entre les éléments d'un document.

La figure 2.7 décrit une grammaire EPF pour la recherche d'un rectangle dans une image. Chaque carré rouge désigne une zone de recherche. Dans cette exemple, nous avons 4 zones nommées *touchUp*, *touchRight* , *touchDown*, *touchLeft*. L'analyseur commence par le recherche d'un segment vertical, ensuite il recherche un segment horizontal passant par la zone *«touchUp»*, après il recherche un segment vertical passant par la zone *«touchRight»*, puis il recherche un segment horizontal passant par *«touchDown»*, enfin l'analyseur que le premier segment vertical trouvé passe aussi par la zone *«touchLeft»*. Une fois la règle est vérifiée, nous pouvons dire que l'image contient le motif *«rectangle»*.

```
rectangle ::= (vLineSeg ---> segLeftSide) &&
    AT(touchUp) && hLineSeg &&
    AT(touchRight) && vLineSeg &&
    AT(touchDown) && hLineSeg &&
    AT(touchLeft) && (vLineSeg <--- segLeftSide)
```

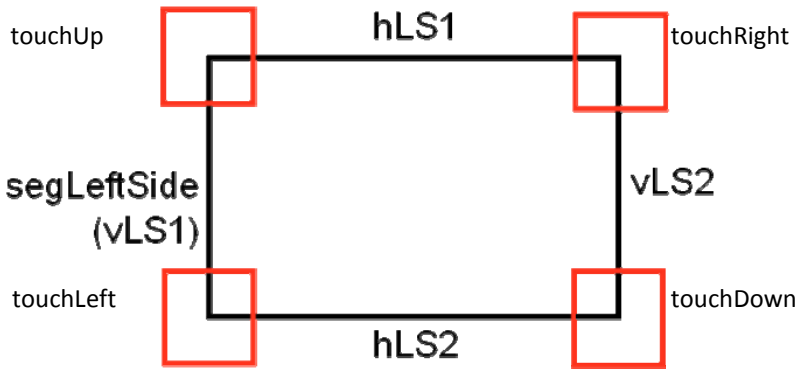


Figure 2.7 – Exemple de règle de grammaire EPF

Les grammaires basées sur les fonctions sont riches en terme d'expressivité car elles sont capables de représenter les différentes relations possibles à travers ses fonctions.

D'autres part, nous pouvons considérer LADDER [76] comme un formalisme à base de fonction. Ce langage ne permet pas seulement la reconnaissance de document mais également la façon dont un document est affiché et édité.

Rappelons que LADDER et GMC-PC offrent une prise en compte forte de la nature manuscrite des données manipulées : LADDER propose un ensemble de seuils prédéfinis par l'utilisateurs tandis que les grammaires GMC-PC se basent sur des calculs statistiques de forme. La modélisation du document nécessite un analyseur pour l'exploiter.

Afin de bien conserver la structure du document, nous choisissons les approches structurelles basées sur les grammaires. Nous considérons que les grammaires basées sur les fonctions pour la description de la relation entre les éléments du document sont les plus adaptées pour notre méthode car elles sont d'une part plus expressives et d'autres part ne sont pas difficiles à implémenter. La gestion des données imprécises est assurée à partir de l'interactivité du système qui va pouvoir solliciter l'utilisateur pour résoudre les problèmes d'ambiguïtés et d'une exploration en largeur des arbres d'analyse permettant de choisir entre plusieurs hypothèses possibles.

L'analyse structurelle à travers une exploration en largeur permettra d'assurer une interactivité avec l'utilisateur en mettant en compétition des hypothèses concurrentes. Nous intégrons des calculs statistiques dans les règles de production afin de bien gérer l'incertitude.

2.5 Analyseur : type d'analyse

Nous nous intéressons dans cette section aux analyseurs associées à des grammaires. Afin de permettre la reconnaissance, il est nécessaire d'avoir une stratégie globale d'utilisation des différentes étapes (extraction des primitive, connaissance a priori, etc.) Au moment où un analyseur analyse l'entrée, il construit un arbre dont les feuilles représentent des instances de symboles terminaux V_T, et à chaque nœud, les différents symboles non-terminaux V_N. La façon dont cet arbre est construit caractérise un paradigme d'analyseur. Selon Belaïd [19], un analyseur peut prendre trois grandes forme : ascendante, descendante ou mixte. Si l'arbre d'analyse est construit à partir de la racine aux feuilles ou l'inverse, l'analyseur est dite *ascendante* ou *descendante* respectivement.

2.5.1 Analyse ascendante

Le principe de l'analyse ascendante est de commencer par extraire les primitives du document, indépendamment des objets à reconnaître. La deuxième phase consiste

à assembler les primitives extraites en introduisant la connaissance pour tenter de reconstruire petit à petit les objets afin de les reconnaitre. L'inconvénient majeur de cette stratégie d'analyse est au niveau de l'extraction de primitives car il n'est pas toujours possible d'extraire les primitives d'une façon indépendante sans aucune information contextuelle. Le retour arrière pour explorer une autre manière d'extraire les primitives peut s'avérer complexe. Parmi les analyseurs ascendant, nous pouvons citer CYK [91].

L'algorithme CYK est un algorithme d'analyse syntaxique conçu pour les grammaires hors-contexte. Comme il est courant pour les algorithmes d'analyse syntaxique, l'algorithme CYK calcule toutes les interprétations syntaxiques possibles de toutes les sous-séquences de la séquence fournie en entrée.

L'efficacité du calcul est fondée sur la propriété suivante :

- Si la grammaire est sous forme normale (une forme est dite normale si et seulement si toutes ses règles de production sont de la forme : $X \rightarrow Y\,Z$ ou $X \rightarrow \alpha$ ou $S \rightarrow \epsilon$ où X, Y, Z ont des symboles non terminaux, α est un symbole terminal, S est l'axiome de la grammaire, et ϵ est le mot vide.), le calcul des interprétations d'une séquence W de longueur l nécessite seulement l'exploration de toutes les décompositions de W en deux sous-séquences exactement.
- Le nombre de paires de sous-séquences à explorer pour calculer les interprétations de W est donc l - 1.

L'idée de l'algorithme CYK est de garder les traces de toutes les analyses de toutes les sous-séquences à l'intérieur d'une table. Le calcul des interprétations syntaxiques se fait ligne par ligne, de bas en haut (c'est-à-dire pour des valeurs croissantes de l). Pour cette raison, on parle d'algorithme de type *ascendant*.

Nous citons également LR(k) [94] que utilise une pile et le flot d'entrée, qui décrivent une configuration de l'analyseur, notée

$$(X_1 ... X_j^1\ a_1 ... a_n^2) \tag{2.1}$$

où les X sont de symboles, terminaux ou non terminaux, stoqués sur la pile, alors que les a sont seulement des symboles terminaux, et correspondent aux terminaux non encore lus sur le flot d'entrée.

2.5.2 Analyse descendante

L'analyse descendante consiste à prédire la présence de primitives dans le document structuré en se basant sur une connaissance a priori, ensuite vérifier leur présence. Il est nécessaire que la connaissance et le contexte soient bien définis pour réussir un tel mécanisme d'analyse. L'inconvénient de cette stratégie est l'explosion de la

combinatoire car l'analyseur peut partir sur des fausses pistes, et donc il n'échoue que tardivement. Dans la littérature, plusieurs analyseurs descendants sont mis en place comme Unger [161]. Shaw [141] présente un analyseur descendant sur la méthode Unger pour PDL. D'autres méthodes d'analyse sont utilisées tel que la descente récursive [73] et LL(k) [73].

Earley [57] analyse la phrase de gauche à droite, de manière prédictive. Cet analyseur ne requière pas une forme particulière de grammaire. Il présente également des inconvénients :

- il n'est pas très intuitif ;
- il n'y a pas moyen de corriger/reconstruire des phrases incorrectes syntaxiquement, ni d'en donner des analyses partielles.

Selon les formalismes grammaticaux présentés auparavant, un certain nombre de méthodes d'analyse ont été développées. Wittenburg et Weitzmann présentent un algorithme d'analyse [169] pour les grammaires relationnelles basé sur l'algorithme de Earley.

Couasnon [48] présente un algorithme d'analyse appelé DMOS basé sur la logique du premier ordre pour analyser des phrases avec un formalisme grammatical EPF.

Nous pouvons souligner que l'analyseur DMOS est caractérisé par trois propriétés principales :

- remise en cause de la structure analysée en cours d'analyse (pour effectuer des segmentations contextuelles) ;
- détection de l'élément suivant à analyser. Contrairement à l'analyseur grammatical classique où l'élément suivant à analyser est l'élément suivant dans la chaÃőne d'entrée, dans l'analyseur bidimensionnel notamment l'analyseur DMOS, l'élément suivant dépend de la position courante du curseur, de la zone de recherche et de l'opérateur de position.
- Gestion correcte du bruit : DMOS offre la possibilité de supporter le bruit grâce à ses préconditions, posticonditions et l'opérateur «*find*». La bonne gestion du bruit est un facteur très important pour le bon fonctionnement d'un analyseur en reconnaissance de documents surtout dans le cas d'une rétroconversion hors ligne.

L'analyse DMOS favorise l'analyse des données en profondeur qu'en largeur, tout en assurant un retour arrière automatique grâce au *Prolog* (l'un des principaux langages de programmation logique).

Les travaux de Marriot [111] et Mace [107] présentent un algorithme d'analyse incrémental analysant les gramamires GMC et GMC-PC.

Le processus d'analyse DALI [107] basé sur le formalisme grammatical GMC-PC présente trois situations différentes (figure 2.8)

- S'il n'y a qu'une interprétation possible, cela signifie que le tracé a été compris sans ambiguïté par le processus d'analyse. La séquence de productions peut donc être appliquée, et le retour visuel correspondant peut être affiché à l'utilisateur.
- S'il y a plusieurs interprétations possibles, cela signifie que le tracé peut être réduit par plusieurs séquences différentes de productions, et donc qu'il y a une ambiguïté. Pour gérer cette ambiguïté, on commence par ordonner les différents degrés d'adéquation, et on prend par la suite les deux meilleurs scores (degré d'adéquation). Si la différence entre ces deux scores est supérieure à un seuil S_a, alors on prend le meilleur score. Dans le cas contraire (la différence est inférieure au seuil), le système supprime le dernier tracé en demandant à l'utilisateur de le redessiner.
- S'il n'y a aucune interprétation possible (degré d'adéquation est inférieur à un seuil minimal), cela signifie au contraire que le tracé n'a pas été compris par le système car aucune production ne peut le réduire. On est donc dans le cas de rejet de distance de tracé. Dans ce cas, on essaie de segmenter le tracé, et on relance le processus d'analyse.

Les techniques LR ont été développées pour analyser les formalismes PG, XPG et SKG dans [42, 43, 41]. Bunke [32] présentent un algorithme d'analyse dérivé de l'algorithme de Earley pour les grammaires Plex hors-contexte. Ce type d'analyse est caractérisé par sa capacité d'exprimer la connaissance.

2.5.3 Analyse mixte

L'analyse mixte consiste à combiner l'analyse ascendante pour extraire les primitives qui ne posent pas de difficultés et l'analyse descendante pour rechercher, à l'aide du contexte, des primitives difficiles à extraire. Les travaux présentés par Golin [70], Jorge [87] et Rekers [134] présentent une méthodologie d'analyse permettant la construction d'arbre d'analyse en deux étapes, à savoir ascendante et descendante. L'étape ascendante construit tous les arbres d'analyse possibles et dans la phase descendante l'algorithme supprime ceux qui ne sont pas valides.

2.5.4 Choix du type d'analyse

L'objectif général de cette thèse est d'élaborer un continuum entre la rétroconversion et la composition. Le choix du type d'analyseur dépend principalement du choix de

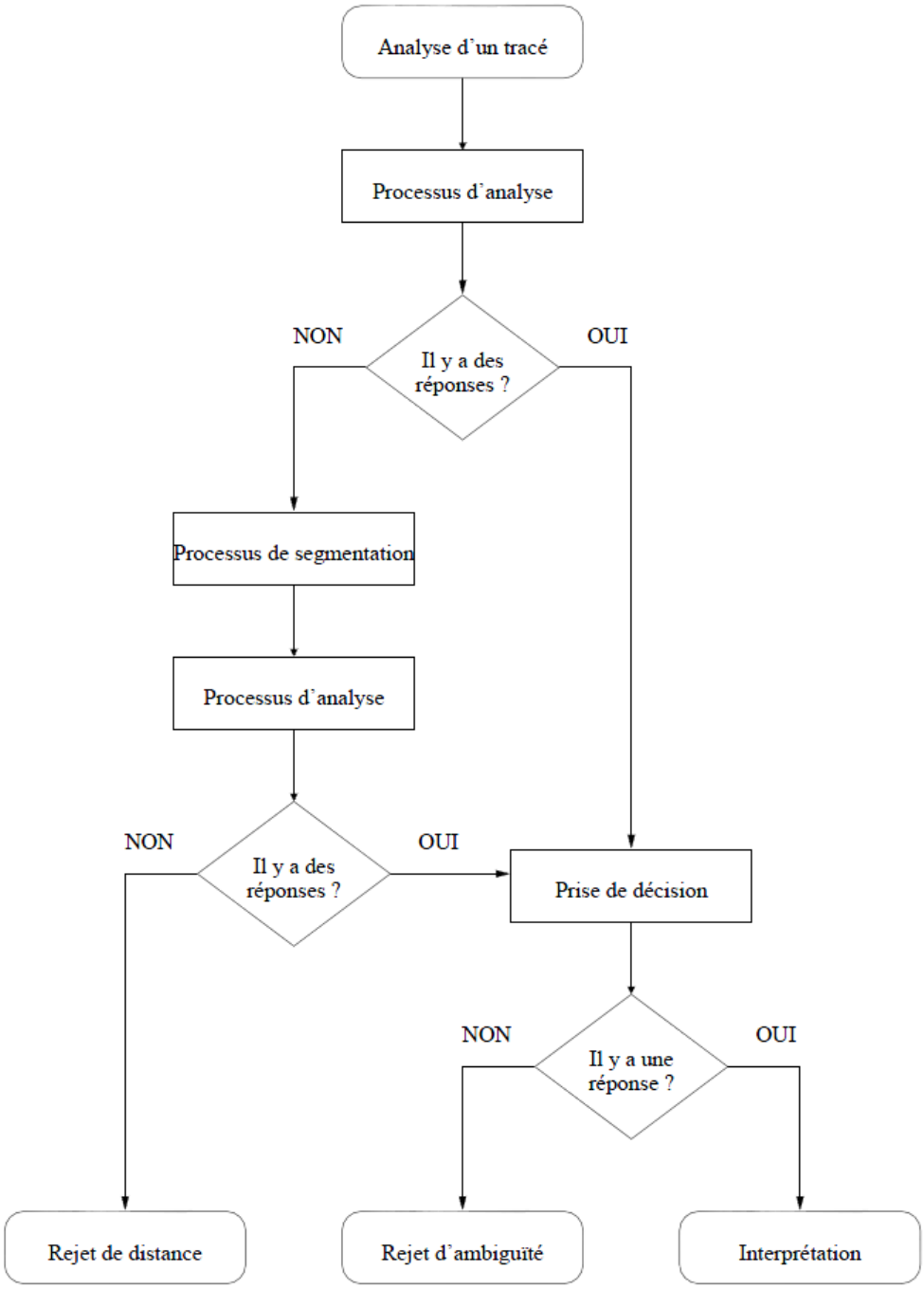


Figure 2.8 – Processus global d'analyse associé à DALI

l'analyseur de la composition/édition. Nous souhaitons un analyseur cohérent entre la rétroconversion et la composition/édition. De plus, il est nécessaire que l'analyseur tient compte de l'interactivité et de la gestion de l'incertitude et de bruit. Nous allons partir de l'analyseur DALI qui est un analyseur descendant basé sur le formalisme grammatical GMC-PC pour la composition des document et nous allons l'étendre pour l'adapter à la rétroconversion. La méthode IMISketch sera donc basée sur le formalisme grammatical GMC-PC et l'analyseur descendant associée.

2.6 Interactivité : interprétation des documents et gestion d'erreurs

L'interactivité des systèmes se caractérise par une partie analyse qui consiste à dégager les informations nécessaires permettant la bonne interactivité et une partie ergonomique qui montre la manière graphique pour présenter l'information. Le côté ergonomique et usage fait l'objet des travaux de thèse à l'équipe LPE de CRPCC [1] du laboratoire LOUSTIC [2]. Dans cette section, nous allons commencer par présenter les principales méthodes interactives et les caractéristiques nécessaires pour bien implémenter une interaction «homme-document».

2.6.1 Méthodes existantes

À notre connaissance, peu de système de rétroconversion introduisent l'interactivité avec l'utilisateur pour la gestion d'erreurs. Les systèmes d'interprétation à la volée sont plus interactifs. Généralement la phase de correction dans un système de rétroconversion n'est pas très détaillée et n'est pas très développée par les chercheurs. Certains systèmes d'interprétation exploitent un mécanisme de correction automatique d'erreurs en fin d'analyse, soit pour attirer l'attention de l'utilisateur sur une interprétation jugée ambigüe soit pour remettre en cause des interprétations déjà validées. Certains travaux se basent sur la recherche de l'incohérence dans le document structuré. L'interaction automatique consiste à solliciter l'utilisateur uniquement en cas de besoin. Plusieurs systèmes se basent sur des seuils de confiance [17, 28, 129]. Ces seuils permettent d'indiquer la validité d'une interprétation et l'ambigüité entre des interprétation. Ces seuils sont parfois appuyés par des calculs statistiques [113].

Gennari [66] propose une technique sémantique pour localiser des erreurs de reconnaissance pour les schémas électriques. Le système utilise la connaissance spécifique du

1. http ://www.sites.univ-rennes2.fr/crpcc
2. http ://www.loustic.net/

domaine lors de l'analyse pour corriger les erreurs de reconnaissance. Par exemple, il indique un éventuel problème de reconnaissance dans un circuit électrique. Il est rare pour un utilisateur de dessiner les connexions de cette façon. L'analyseur suggère ainsi qu'un composant a été manquée. Une autre indication est la présence d'une incohérence entre un composant électrique et le nombre de connexions qui y associées.

Blostein [25] détecte les erreurs effectuées par un système d'interprétation de partitions musicales en recherchant l'incohérence au niveau de la durée des notes constituant une mesure avec la durée théorique de cette mesure, etc. Liu [166] a proposé une interactivité basée sur un algorithme générique pour la reconnaissance de graphiques [167, 168]. L'utilisateur peut spécifier un ou plusieurs exemples d'un type d'objets graphiques dans dans un document structuré, le système alors apprend et reconnaître des objets similaires dans les le même document ou des documents similaires.

Mankoff [110] propose une interaction avec l'utilisateur dans les cas d'ambigüités. Le but de leurs travaux n'est pas la détection des cas d'erreurs potentiels, même si un certain nombre de pistes sont évoquées (par exemple la détection d'ambigüités fréquentes à l'aide de matrices de confusion). Le système sollicite l'utilisateur s'il considère plusieurs façon d'interpréter un groupe de primitives. Il s'agit de trouver des techniques dite *médiation* permettant au système de solliciter l'utilisateur d'une manière conviviale en cas de besoin. Un bon médiateur minimisera l'effort nécessaire à l'utilisateur pour corriger les erreurs ou pour sélectionner l'interprétation correcte. Notons que des recherches ont prouvé qu'un bon interfaçage entre le système et l'utilisateur, menant à une correction intuitive des erreurs, peut réduire certains aspects négatifs des erreurs de reconnaissance.

Notowidigdo [123] utilise dans sa méthode de rétroconversion hors-ligne une médiation qui affiche les différentes interprétations possibles pour un symbole sélectionné par l'utilisateur. La détection des erreurs dans cette méthode n'est pas gérée par le système d'interprétation mais plutôt par l'utilisateur. Il utilise une interface de médiation qui présente d'autres interprétations des parties de croquis. Cette interface est affichée lorsque l'utilisateur sélectionne une forme qui pourrait avoir d'autres interprétations.

2.6.2 Caractéristiques de l'interaction «homme-document»

Le but de notre thèse est d'élaborer une méthode de rétroconversion permettant de solliciter l'utilisateur en cas de besoin afin d'éviter la propagation d'erreur et donc une phase de vérification qui pourrait être fastidieuse. Nous résumons dans la suite de cette section les critères de base tels que la présentation des hypothèses d'interprétation, le moment de sollicitation, le contexte, l'interactivité pour intégrer une interface Homme-

Document dans un logiciel. Ces critères entreront dans le choix entre deux ou plusieurs interprétations possibles.

2.6.2.1 Le moment de la sollicitation

Le moment de la sollicitation est un facteur important de l'interactivité du système. Plusieurs variantes ont été proposées dans la littérature. Certains travaux proposent une sollicitation à la fin de l'analyse [85, 69]. Malheureusement, nous considérons que cette stratégie favorise une propagation d'erreurs et donc une phase de correction qui pourrait être fastidieuse. Une deuxième alternative consiste à mettre en pause le système dans le cas de détection d'erreurs par l'utilisateur. Cette alternative demande un suivi détaillé de l'analyse par l'utilisateur pour ne pas laisser la propagation d'erreurs.

Une autre possibilité qui consiste à solliciter automatiquement l'utilisateur durant la phase d'analyse est envisageable. Si plusieurs hypothèses d'interprétations sont possible, l'analyseur est capable de solliciter l'utilisateur. Cette alternative permet d'éviter la propagation des erreurs mais pose des contraintes tel que le temps d'analyse, le nombre de sollicitation utilisateur, etc.

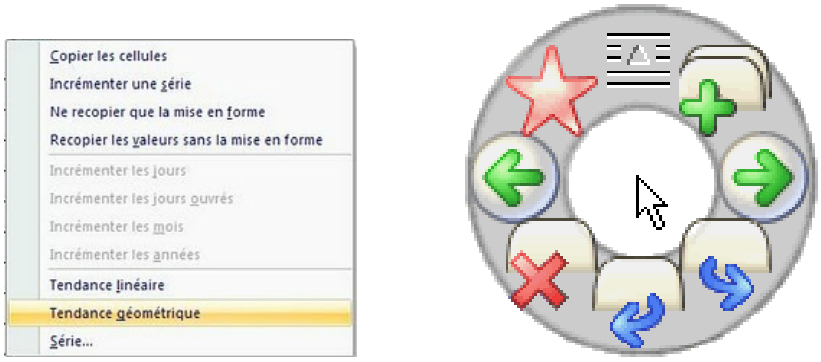
2.6.2.2 La présentation des hypothèses d'interprétation

Elle consiste à décrire la façon dont les hypothèses seront présentées à l'écran à l'utilisateur. La présentation d'une hypothèse à l'utilisateur pourrait nécessiter des informations supplémentaires décrivant le contexte pour bien choisir la bonne hypothèse. Le choix des informations à présenter à l'utilisateur reste un challenge scientifique. Un manque d'information pourrait engendrer une fausse interprétation par l'utilisateur. Une surcharge d'information engendre une présentation trop chargée.

Une présentation de toute les hypothèses à la fois sur l'écran offre à l'utilisateur la possibilité de valider rapidement la bonne interprétation. L'inconvénient d'une telle présentation est la surcharge de l'interface de sollicitation. Une présentation séquentielle qui affiche à l'utilisateur hypothèse par hypothèse évite cette surcharge mais elle est plus coûteuse surtout si l'utilisateur valide la dernière hypothèse.

Généralement les auteurs choisissent des menus linéaires (figure 2.9(a)). D'autres auteurs préfèrent des menus circulaires [98] (figure 2.9(b)) et des menus sous forment d'une grille [15, 148]. De plus, certains travaux présentent présentent les hypothèses d'interprétation possibles sous forme d'un dessin localisé à l'endroit de l'ambigüité.

Ce choix relève davantage du domaine de la psychologie cognitive. Nous serons assistés par les chercheurs du CRPCC pour répondre à ce choix.



(a) Interface linéaire (b) Interface circulaire

Figure 2.9 – Exemple d'interface présentant plusieurs hypothèses possibles

2.7 Discussion et choix du formalisme

Le choix des primitives et la méthode d'analyse sont fortement dépendantes. Un ensemble de primitives riches pour simplifier leur regroupement. Dans certains cas, les systèmes de reconnaissance sont sensible à la sur-segmentation. Cette sur-segmentation n'est pas toujours facile à l'implémenter. Par contre, le choix des primitives basique engendre une sous-segmentation et une explosion combinatoire.

Dans notre méthode IMISketch, nous allons choisir des primitives basiques, les segments, et nous allons proposer des techniques pour éviter l'explosion combinatoire.

À partir des sections 3.1.2.1 et 3.1.2.2, nous constatons que d'une part les méthodes statistiques offrent généralement de meilleures performances en terme de reconnaissance de formes, essentiellement lorsque les données d'entrée sont manuscrites. D'autre part, les méthodes structurelles permettent de bien modéliser et décrire la structure de document. Notons que quelques méthodes de reconnaissance tel que DALI [108] sont des méthodes *hybrides* qui exploitent une combinaison entre l'approche structurelle et l'approche statistique.

L'objectif du projet ANR est d'aboutir à un continuum entre un document technique sous sa forme papier et ce même document sous sa forme numérique interprétée. Le principe est de reconnaitre un document existant numérisé ou vectoriel pour en extraire une représentation numérique interprétée et pouvoir ensuite l'éditer, en contexte de mobilité terrain, à travers une modalité d'interaction «homme-document» orientée stylo. Ce travail ne consiste pas uniquement à rétroconvertir des documents structurés mais aussi à pouvoir les éditer et les compléter. Nous allons étendre la méthode DALI [108] basée sur les grammaire GMC-PC (plus de détail dans la section 3) pour la manipulation de document après sa rétroconversion. Cette extension consiste à résoudre les problèmes liés à l'interprÃĪtation a posteriori tel que la gestion de la combinatoire.

Nous avons choisi également un analyseur descendant car il exprime bien la connaissance et permet aussi d'avoir une cohérence avec le moteur d'analyse *DALI* (analyseur descendant) ce qui facilitera le développement du continuum.

Deuxième partie

Principes spécifiques : Grammaires de multi-ensembles à contraintes pilotées par le contexte

CHAPITRE 3

Grammaires de multi-ensembles à contraintes pilotées par le contexte

Ce chapite est inspiré de la thèse de Macé [108].

Dans la section 2.4.3.2, nous avons présenté les grammaires les plus reconnues pour la description des documents structurés. Cependant, nous avons également mis en avant plusieurs de leurs limitations.

Nous avons constaté que certains formalismes sont envisageables pour le rétroconversion des documents structurés de différentes natures (imprimés, manuscrits et vectoriels). Le but de cette thèse n'est pas uniquement d'élaborer une méthode de rétroconversion mais aussi d'assurer un passage fluide et cohérent depuis un document non interprété vers son format numérique reconnu afin de pouvoir l'éditer et le compléter. Ce continuum évoque des contraintes supplémentaires à nos travaux dont notamment le choix du formalisme grammatical. L'utilisation d'un formalisme commun entre la phase de rétroconversion et la phase de composition/édition facilitera la mise en place du continuum. Nous avons donc choisi les *grammaires de multi-ensembles à contraintes pilotées par le contexte* (GMC-PC) qui ont été conçues pour la composition *en-ligne* à la volée des documents structurés.

Nous allons partir de ce formalisme pour l'étendre à la problématique de la rétroconversion interactive de document qui fait l'objet de ce travail de thèse. Avant cela, nous présentons dans ce chapitre le formalisme initialement conçu pour la composition à la volée de document en mettant en avant ses limitations vis à vis de notre problèmatique de rétroconversion a posteriori interactive.

Nous commençons ce chapitre par une description formelle des GMC-PC, ensuite nous décrivons les limites de ces grammaires, par rapport à la rétroconversion. Enfin,

nous réalisons un bilan concernant le nouveau formalisme que nous allons créer, en mettant en avant les extensions envisagées

3.1 Description formelle des GMC-PC

Le formalisme GMC-PC (grammaire de multi-ensembles à contraintes pilotée par le contexte) est générique et formé par un 4-uplet G = (V_N, V_T, S, P), avec

- V_T est un *alphabet* : ensemble de symboles terminaux ;
- V_N est un ensemble de symboles non terminaux, avec $V_T \bigcap V_N = \emptyset$;
- S est *l'axiome* : le symbole de départ, avec $S \in V_N$;
- P est un ensemble de règles de productions, ou plus simplement appelées *production* (appelées aussi des règles de ré-écriture) ;

Chaque production $p \in P$ est de la forme :

$$\alpha \rightarrow \beta \left\{ \begin{array}{l} Préconditions \\ Contraintes \\ Postconditions \end{array} \right\} \text{ et } (D)$$

avec $\alpha \in V_N^+$ et $\beta \in (V_N \bigcup V_T)^+$

Les grammaires GMC-PC sont conçues pour l'interprétation des document en-ligne à la volée. Une production GMC-PC consiste à remplacer un multi-ensemble d'éléments β par un multi-ensemble d'éléments α si un certain nombre de conditions sont satisfaites, D modélisant les attributs des éléments de α. Contrairement à la majorité des approches existantes basées sur une modélisation des symboles du document en termes de primitives graphiques structurelles prédéfinies, telles que le segment, l'arc, le cercle, etc, Macé [108] propose de définir l'alphabet à une seule primitive graphique, qui est le tracé manuscrit, c'est-à-dire la séquence de points entre un poser et un lever de crayon. Il considère qu'avoir une seule primitive graphique est un avantage surtout quand il s'agit d'une forme complexe.

$$V_T = \{Trace\}$$

Une règle de production GMC-PC est constituée de trois parties : préconditions, contraintes et postconditions. Les préconditions et les postconditions modélisent *une vision globale* du document. Tandis que les contraintes modélisent *une vision locale* des éléments analysés. La figure 3.1 illustre un exemple de règle de production GMC-PC pour la composition d'un mur dans un document de plan d'architecture. Le tracé t se transforme en un mur $mRes$ si les préconditions et les contraintes sont satisfaites. Les

préconditions vérifient si le tracé t passe par l'extrémité du mur $m1$. Si les préconditions sont vérifiées, l'analyseur vérifie les contraintes. Dans cet exemple, il faut que le tracé t soit assez long (la contrainte traceLong(t)) et il ressemble à un segment (reconnaisseurSegment(t,Mur)). Une fois ces contraintes vérifiées, la règle est appliquée et le tracé t est transformé en un mur. Le mur créé pilotera l'analyse via les postconditions. Dans cet exemple, le passage du tracé $tn1$ par la zone «extrémité» déclenche la règle qui transforme *un tracé* en *un mur*. Si le tracé $tn2$ est contenu dans la zone «milieu», l'analyseur déclenche la règle de production qui transforme *le tracé* $tn2$ en *un porte*.

Mur : mRes $\rightarrow$ **Tracé : t** avec {

Préconditions :

$\{(Mur : m1)[extrémité]t[un]\}$

Contraintes :

traceLong(t) &
reconnaisseurSegment(t, Mur)

Postconditions :

$\{mRes[extrémité](Tracé : tn1)[un] \Rightarrow [Mur \rightarrow tn1]\}^*$ &
$\{mRes[milieu](Tracé : tn2)[tous] \Rightarrow [Porte \rightarrow tn2]\}^*$

Figure 3.1 – Production GMC-PC pour la composition d'un mur

3.1.1 Vision globale du document

Dans le formalisme GMC-PC, la vision globale décrit les éléments mis en jeu par une production en se basant sur le positionnement relatif des éléments du document. Cette modélisation est introduite par des *Contextes Structurels de Documents*, ou CSD. Un CSD est une contrainte spécifique constituée d'une part d'un emplacement dans le document, et d'autre part (de sous-parties) d'éléments qui sont attendus à cet emplacement.

3.1.1.1 Syntaxe

La syntaxe d'un CSD est la suivante :

$$\gamma[position]\delta[partie]$$

avec :

- γ est un ensemble d'éléments dits *références* ;
- $[position]$ est *un opérateur de position*, désignant ainsi une position relativement à ces références ;

- $\gamma[position]$ décrit un emplacement dans le document ;
- δ est un ensemble d'éléments dits *attendus* ;
- $[partie]$ est un opérateur de sélection, désignant une partie des points de chacun des éléments de δ (e.g. *tous* leurs points, leur *premier* point, leur point le plus à *gauche*, *un* de leur points, etc.) ;
- $\delta[partie]$ représente un sous-ensemble des points constituant les attendus.
  Par conséquent, on dit qu'un CSD est satisfait si, à l'emplacement $\gamma[position]$ on trouve bien les points $\delta[partie]$.

La séparation entre l'emplacement dans le document et ce qui est attendu dans cet emplacement va permettre la mise en œuvre d'une stratégie d'analyse par *prédiction-vérification*.

La figure 3.2 illustre la notion de vérification de CSD à l'aide d'un exemple du domaine des plans d'architecture. Dans le document présenté, il existe un mur, noté m1 et deux tracés manuscrits, notés t1 et t2. La figure présente un emplacement de CSD ((m1)[milieu]), représentés à l'aide d'un rectangle pointillé. Dans cet exemple, le CSD

$$(m1)[milieu](t1)[tous]$$

indique que le tracé $t1$ doit être contenu en totalité dans la zone *milieu* du mur $m1$. Ce CSD est bien vérifiée alors que

$$(m1)[milieu](t2)[tous]$$

n'est pas vérifiée. Les CSD sont utilisées aussi bien dans les postconditions que dans les préconditions qui jouent le rôle de prédiction-vérification dans les GMC-PC.

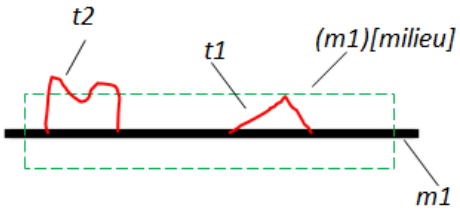


Figure 3.2 – Notion de vérification de CSD

3.1.1.2 Préconditions

Une précondition est un ensemble de CSD qui doivent être satisfaits, et donc qui doivent avoir été déclenchés par les éléments de la partie droite de la production. Elle

vérifie que les éléments considérés (pour lesquels on teste s'ils peuvent correspondre à la partie droite de la production) sont dans un contexte structurel cohérent avec la production. Une précondition référence donc l'ensemble des CSD, créés auparavant en postconditions, qui doivent être satisfaits par ces symboles.

La propriété majeure des CSD en précondition est de rendre les productions sensibles au contexte. Notons que les CSD en préconditions permettent de référencer des symboles qui ne sont pas réduits par la production (les références associées à ces CSD) et de les utiliser dans les blocs suivants de cette production.

En reprenant l'exemple de la production d'un mur, le bloc de précondition est définie comme suit :

Préconditions :

$$\{(Mur : m1)[extrémité]t[un]\}$$

Ce bloc de précondition vérifie si le tracé t de la partie droite de la production passe par la position *extrémité* d'un mur déjà interprété.

Si la précondition d'une production est satisfaite, alors nous pouvons dire que la vision globale est bien vérifiée et l'analyseur du formalisme passe à la vérification de la vision locale assurée par les contraints. Notons que la vision globale (précondition et postcondition) ne prend pas directement en compte la forme des éléments, même si la forme des opérateurs de position des CSD peut d'une certaine manière la modéliser partiellement.

3.1.1.3 Postconditions

Le rôle des postconditions est de mettre à jour les informations structurelles qui découlent de l'analyse courante et qui permettront l'analyse de la suite du document. Elles modélisent quels symboles peuvent maintenant exister dans le document, en conséquence de la réduction de la production. La règle de production pilote la suite de l'analyse via les postconditions en utilisant les CSD. La forme général d'un CSD utilisé en postcondition est alors :

$$\{\gamma[position]\delta[partie] \Rightarrow [\alpha \rightarrow \beta]\}^q$$

avec :

- γ est un ensemble d'éléments de document ;
- δ est un multi-ensemble de classe de symboles pouvant exister dans le document $\delta \subseteq V_N^+$;

- α, β sont des multi-ensembles d'éléments ou de classes de symboles, $\alpha \rightarrow \beta$ modélisant par conséquent *un format de production* ;
- q est un entier qui indique le nombre de fois que ce contexte doit être vérifié pour qu'il soit rempli.

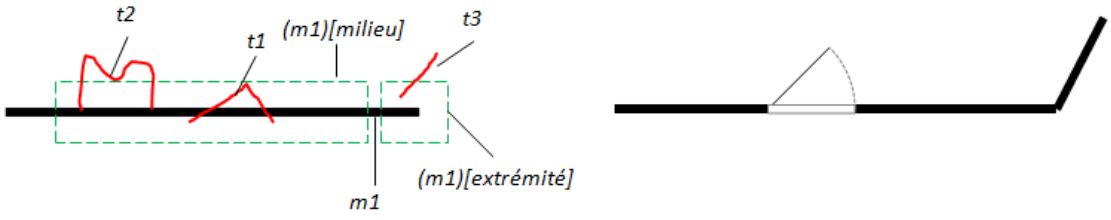
Un CSD des postconditions modélise qu'à une zone [*position*] créée par l'élément γ, on attend le sous-ensemble de points [*partie*] de symboles de types précisés par δ. Si la [*partie*] de δ est bien dans la zone [*position*] de l'élément γ et satisfait le CSD, l'analyseur déclenche toutes les productions de la forme $\alpha \rightarrow \beta$. En plus de l'exploitation du contexte pour piloter l'analyse, les postconditions modélisent l'ordre dans lequel ces symboles vont être composés, puisque les actions modélisées par le format de production ne peuvent être effectuées qu'après la réduction de la production courante.

La figure 3.3 illustre le rôle des postconditions sur un exemple de plans d'architecture. La règle de production permettant la création d'un mur en occurrence $m1$ exploite les mêmes postconditions présentées dans la figure 3.1.

Postconditions :

$$\{mRes[extrémité](Tracé : tn1)[un] \Rightarrow [Mur \rightarrow tn1]\}^* \,\&$$
$$\{mRes[milieu](Tracé : tn2)[tous] \Rightarrow [Porte \rightarrow tn2]\}^*$$

Cette production présente la localisation des nouveaux CSD correspondants (figure 3.3(a)) ; elle montre également un exemple de suite de document exploitant ces contextes (figure 3.3(b)). Le tracé $t1$ et $t3$ respectent bien le CSD des postconditions donc ils sont transformés en une porte et un mur respectivement. Alors que le tracé $t2$ n'est pas vérifié par le CSD donc il est rejeté.



(a) Ajout des trois tracés $t1$, $t2$ et $t3$ sur un mur déjà interprété

(b) Interprétation des tracé $t1$ et $t3$. Pas de règle vérifiée pour l'interprétation du tracé $t2$

Figure 3.3 – Exemple de postconditions d'une production GMC-PC

Dans cette partie, nous avons montré comment les postconditions permettent de maintenir à jour la structure physique globale du document. Le pilotage par le

contexte permet de réduire le nombre d'interprétations à considérer. Par conséquent, la contrainte résultante de l'ordre de composition imposée à l'utilisateur permettra d'obtenir des performances intéressantes, à la fois en termes de temps de calcul que de reconnaissance de formes.

3.1.2 Vision locale des éléments interprétés : les contraintes

Les GMC-PC modélisent une vision locale des éléments de la partie droite de la production. Cette vision locale est introduite à partir de la reconnaissance structurelle de forme et la reconnaissance statistique de forme. Elle est décrite dans la partie **contraintes** du formalisme. Les contraintes peuvent avoir deux niveaux :

- *contraintes structurelles* qui consistent à vérifier l'agencement relatif des constituants du symbole considéré ;
- *contraintes statistiques* qui consistent à reconnaître les formes complexe peu structurées et pas facile à décrire au niveau structurel.

3.1.2.1 Contraintes structurelles

Les contraintes de reconnaissance structurelle visent à modéliser l'agencement relatif des constituants du symbole. Elles sont fortement adaptées pour la représentation des symboles fortement structurés. Dans une production GMC-PC, plusieurs contraintes structurelles sont possibles. En revenant à l'exemple de production illustré dans la figure 3.1, nous avons la contrainte structurelle $traceLong(t)$. Cette contrainte vérifie le longueur du tracé afin de valider la possibilité de le transformer en un mur.

3.1.2.2 Contraintes statistiques

Les contraintes statistiques sont généralement utilisées pour reconnaître les symboles peu structurés ou bruités. Ces contraintes sont capables de manipuler les données imprécises. Comme les méthodes statistiques de reconnaissance de formes, les contraintes statistiques exploitent des classifieurs basés sur les modèles de Markov cachés, des réseaux de neurones ou des séparateurs à vastes marges (SVM). Dans l'exemple illustré dans la figure 3.1, $reconnaisseurSegment(t, Mur)$ désigne que la production utilise un classifieur nommé $reconnaisseurSegment$ afin de vérifier si le tracé t ressemble à un Mur.

3.2 Évaluation d'une production GMC-PC

Les grammaires GMC-PC exploitent la théorie des sous-ensembles flous afin d'évaluer le degré d'adéquation entre un ensemble d'éléments de la partie droite β d'une production et une interprétation de ces éléments de la partie gauche α de cette même production. Il ne s'agit pas d'une évaluation discriminante de cette interprétation ce qui est adapté au traitement des données imprécises. Chaque production GMC-PC aura un degré d'adéquation ρ avec $\rho \in [0, 1]$ entre β et α, c'est-à-dire le degré avec lequel la production est satisfaite. Le degré d'adéquation dans le formalisme GMC-PC est défini à partir des préconditions et des contraintes.

3.2.1 Degré d'adéquation des préconditions

Les préconditions mesurent l'adéquation entre le contexte structurel dans lequel des symboles sont localisés, et le contexte dans lequel ils devraient effectivement se trouver pour que la production soit applicable. Ce degré est déduit des CSD qui les constituent en s'appuyant sur les notions de positionnement relatif flou [24]. Il s'agit donc d'une part de positionner l'emplacement $\gamma[position]$ dans le document, indépendamment des éléments effectivement attendus. D'autre part, il faut évaluer le degré avec lequel les points modélisés par $\delta[partie]$ sont situés à cet emplacement. Ceci modélise le degré d'adéquation d'un CSD.

Chaque position [*position*] utilisée dans la grammaire associe une fonction d'appartenance $\mu_{position}$, appelée *paysage flou*. Un paysage flou définit un sous ensemble flou de l'espace du document. Le terme $\mu^{\gamma}_{position}(E)$ désigne le degré d'appartenance d'un objet E à un contexte structurel localisé à la position $\gamma[position]$. Plusieurs techniques peuvent être utilisées pour positionner un paysage flou [108].

L'évaluation de l'appartenance d'un objet E à une *position* dépend de l'opérateur de sélection *partie*. Le degré d'appartenance dépend de l'opérateur de sélection :

- Si l'opérateur met en jeu un point particulier de l'élément E (par exemple : *premier*, *dernier*, etc), le degré d'appartenance de E au paysage flou est alors le degré d'appartenance du point considéré au paysage flou.
- Si l'opérateur est *un*, c'est à dire au moins un point doit intersecter le paysage flou, le degré d'appartenance de E est alors le degré d'appartenance de son point qui le vérifie le mieux :

$$\mu^{\gamma}_{position}(E) = \frac{A}{|E|} \underset{x \in E}{max}\ \mu^{\gamma}_{position}(x) \tag{3.1}$$

- Si l'opérateur est *tous*, c'est à dire au moins tous les points doivent intégrer le paysage flou, le degré d'appartenance de E est alors la valeur moyenne de

l'évaluation globale de l'appartenance de E au sous-ensemble flou $\mu^{\gamma}_{position}$:

$$\mu^{\gamma}_{position}(E) = \frac{A}{|E|} \sum_{x \in E} \mu^{\gamma}_{position}(x) \tag{3.2}$$

Afin d'évaluer le degré $\mu^{\gamma}_{position}(\delta)$ avec lequel un CSD est globalement satisfait, le formalisme GMC-PC se base sur un combinaison floue de l'appartenance de chacun des éléments appartenant à δ. Il opte le T-norme probabiliste (équation 3.3)

$$mu^{\gamma}_{position}(\delta) = [\prod_{E \in \delta} \mu^{\gamma}_{position}(E)]^{\frac{1}{|\delta|}} \tag{3.3}$$

Une fois les degrés d'appartenance des CSD des préconditions sont établis, le formalisme déduit le degré de vérification des préconditions en fusionnant ces degrés à travers l'opérateur de fusion T-norme probabiliste.

3.2.2 Degré d'adéquation des contraintes

L'évaluation de la vérification des contraintes $\mu_{contraintes}$ est déduite à partir d'opérateur de fusion T-norme probabiliste avec normalisation appliqué à toutes les contraintes (structurelle et statistique) modélisées par le concepteur dans une règle de production GMC-PC. En terme de de système de reconnaissance, les GMC exploite des systèmes d'inférence flous [10, 26, 153].

3.2.3 Déduction du degré d'adéquation d'une production et d'une interprétation

Le degré d'adéquation d'une production ρ_p mesure l'adéquation des éléments de la partie droite β d'une production aux éléments de la partie gauche α. Il est déduit à partir d'une fusion des degrés des préconditions et conditions. Cette fusion est présenté dans l'équation 3.6.

$$\rho_p = \sqrt{\mu_{préconditions} \cdot \mu_{contraintes}} \tag{3.4}$$

Une interprétation est une séquence de productions. Le degré d'adéquation d'une interprétation ρPS formée par PS productions est calculé en se basant aussi sur la T-norme géométrique normalisé par le nombre de productions de l'interprétation (équation 3.5).

$$\rho PS = (\prod_{P_i \in PS} \rho_{p_i})^{\frac{1}{|PS|}} \tag{3.5}$$

avec $|PS|$ désigne le nombre de production de l'interprétation PS considérée.

3.3 Techniques de rejet

La théorie de la logique floue offre la possibilité de mettre en compétition les productions qui ne sont pas exactement vérifiées, ce qui est essentiel pour la manipulation des données imprécises. Le formalisme GMC-PC dispose des techniques de rejets de distance (ou rejet d'ignorance) [119] pour réduire l'espace de recherche et éviter alors l'explosion de la combinatoire engendré par la théorie de la logique flou en élaguant les interprétations invalides. Une interprétation est dite invalide si seulement si l'un des degrés des CSD des préconditions et contraintes est inférieur à un seuil T_d défini a priori.

La prise de décision consiste alors à choisir l'interprétation qui a le degré d'adéquation le plus élevé. Le formalisme GMC-PC propose également un rejet d'ambiguïté [119] dans le cas des interprétations ayant des degrés d'adéquations ρ très proches. La détection d'une ambiguïté entre les interprétations PS_1 et PS_2 de degrés respectifs ρPS_1 et ρPS_2 avec $\rho PS_1 \geq \rho PS_2$ est donnée comme suit :

$$\Psi PS_1, PS_2 = \frac{\rho PS_1 - \rho PS_2}{\rho PS_1} \tag{3.6}$$

Si $\Psi PS_1, PS_2$ est inférieur au seuil d'ambiguïté T_a, nous pouvons dire que les interprétations sont ambigues et donc l'analyseur rejette les deux séquences de productions.

3.4 Limitation de GMC-PC

Les grammaires GMC-PC ont été conçues pour la reconnaissance de document structuré *en ligne à la volée*. Ce formalisme offre des avantages en terme de couplage statistique et structurelle, de gestion de l'incertitude à travers la logique flou et de généricité. Cependant, ce formalisme a des limites que nous allons discuter dans cette section.

Les GMC-PC sont caractérisées par la forte prise en compte du contexte temporel. En effet, il est nécessaire de terminer un symbole avant de passer au suivant. Ce choix stratégique a pour but de réduire la combinatoire. Cependant, si l'on veut utiliser ce formalisme pour la rétroconversion de documents, ces contraintes temporelles doivent être ignorées car le document contient toutes ses constituants avant la phase d'analyse.

De plus les GMC-PC consistent à interpréter une primitive (en occurrence un tracé) dans un document partiellement reconnu alors que la rétroconversion consiste à interpréter des primitives dans un contexte de document partiellement interprété. En rétroconversion, toutes les primitives peuvent être interprétées dans tous les sens, ce qui engendre facilement des explosions combinatoires.

En terme d'interactivité, le formalisme est conçu uniquement pour une interprétation à la volée avec un retour visuel du résultat de l'interprétation de chaque tracé. L'expolitation du formalisme ne gère pas l'interactivité avec l'utilisateur.

3.5 Bilan

Dans ce chapitre, nous avons présenté le formalisme grammatical GMC-PC mis au point pour la composition à la volée des documents structurés. Vu que l'objectif de nos travaux est d'aboutir à un continuum entre un document technique sous sa forme papier et ce même document sous sa forme numérique interprétée, la tâche principale consiste à rétroconvertir les documents pour pouvoir ensuite les éditer et les compléter. Le choix d'étendre le formalisme des GMC-PC au domaine de la rétroconversion interactive est stratégique car il garantit une cohérence dans la modélisation de document que ce soit en composition ou en rétroconversion. Les extensions de ce formalisme vont porter notamment sur l'analyseur qui devra faire face au surcoût de combinatoire engendrée par la retroconversion d'un document, mais aussi à la nature du signal d'entrée à traiter ou encore à l'interactivité qui devra être pilotée par l'analyseur.

Nous allons présenter, dans le chapitre suivant, notre méthode interactive IMIS-ketch pour la rétroconversion des documents structurés. Cette méthode basée sur les grammaires GMC-PC, est capable d'interpréter les documents structurés de différentes natures (manuscrit, imprimé et vectoriel), de manière interactive.

Troisième partie

Contribution : la méthode IMISketch

CHAPITRE 4

Description de la méthode IMISketch

Dans ce chapitre, nous détaillons les différentes parties de la méthode interactive *IMISketch*. *IMISketch* sollicite l'utilisateur en cas de besoin afin de réduire une phase de vérification qui pourrait être fastidieuse. La méthode *IMISketch* est constituée de quatre grands blocs illustrés sur la figure 4.1 : 1) un bloc de prétraitement qui a pour but d'extraire les primitives du document à rétroconvertir, 2) un bloc modélisant la connaissance associée aux documents à reconnaître, 3) un bloc gérant la construction des arbres d'analyse et permettant l'exploration des interprétations possibles et 4) un bloc de prise de décision qui valide la bonne interprétation soit *implicitement* de façon automatique soit *explicitement* en sollicitant l'utilisateur. Nous allons détailler dans les sections suivantes les quatre principaux blocs du processus d'analyse d'IMISketch. Le bloc de prétraitement est le seul bloc indépendant des autres blocs. Le bloc de la connaissance a priori est sollicité par le bloc de construction des arbres et le bloc de la prise de décision. En fait, la construction des arbres d'analyse correspond à l'application des règles de productions modélisées par la grammaire et la prise de décision calcule les scores des hypothèses en se basant sur la connaissance a priori (grammaire et classifieur). Le bloc de construction des arbres est fortement dépendant du bloc de prise de décision. En effet, chaque nœud créé par le bloc de construction a un score déterminé à partir du bloc de prise de décision.

4.1 Prétraitement : extraction des primitives

Le choix des primitives mises en entrée du système est lié au type de document à reconnaitre. Nous étudions brièvement les choix faits dans la littérature concernant les plans d'architecture. La reconnaissance de plans architecturaux a déjà été envisagée notamment par Dosch et al dans [55]. Dans ces travaux, les segments de droite, représentatifs des murs, sont des éléments primitifs très largement utilisés. La plupart des

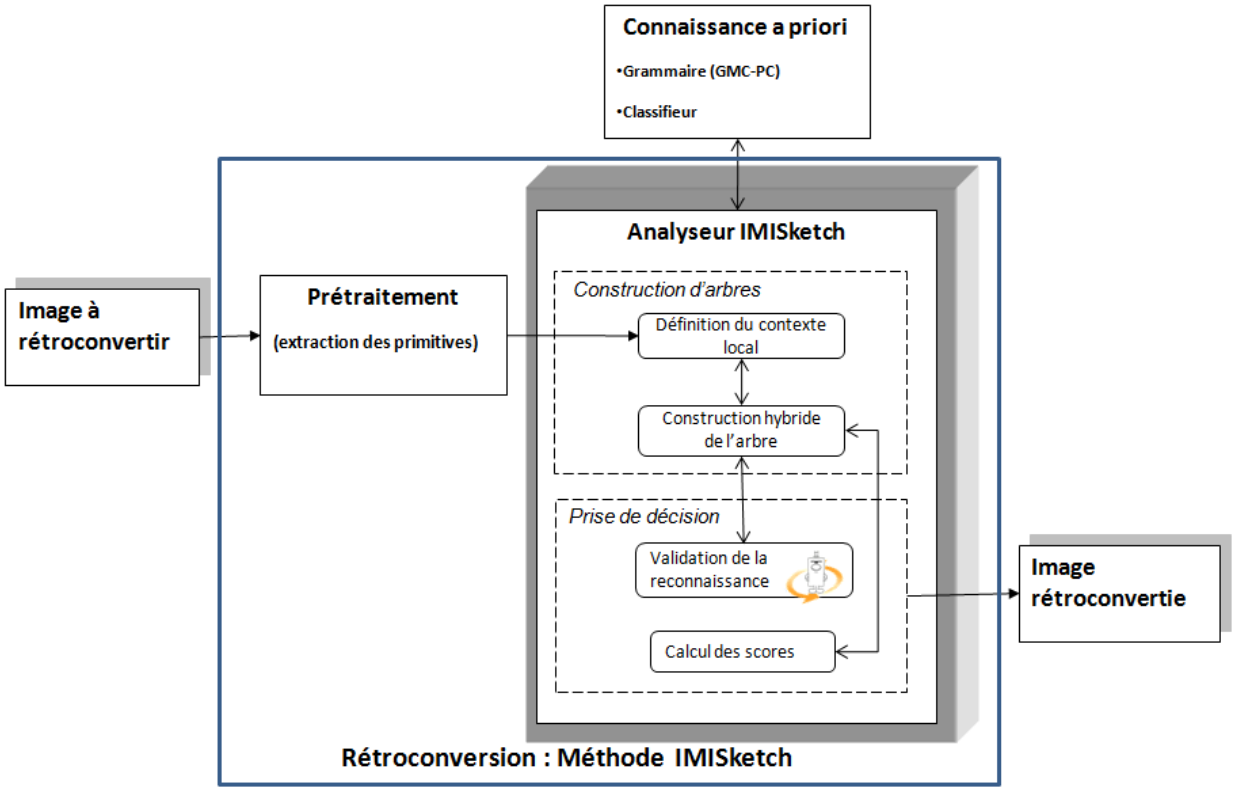


Figure 4.1 – Processus global d'analyse associé à IMISketch

plans analysés ont été dessinés à la règle (ou avec un logiciel de CAO) et ces segments sont réellement rectilignes. Hilaire propose dans [80] une autre approche améliorant celle de Dosch. L'originalité de ses travaux se présente essentiellement au niveau du processus de segmentation du document. Cet aspect est jugé important surtout lorsque les symboles s'intersectent dans un document. Hilaire, comme Dosch, s'intéressent aux documents imprimés, alors que notre but est d'extraire les primitives de documents de différentes natures (manuscrit, imprimé, etc). Ainsi, les murs peuvent le plus souvent être approximés à des segments de droite, alors que la représentation des portes et des fenêtres fait parfois appel à des arcs de cercle très grossièrement dessinés.

Dans [35], Chang propose de réaliser la vectorisation de documents tracés à la main par des courbes de Bézier. Cette approche a été conçue pour la vectorisation des bandes dessinées, par contre elle ne conduit pas à une représentation facile à utiliser pour une interprétation ultérieure du tracé en tant que pièces, portes, fenêtres,... Dans [51] de Brucq utilise un filtre de Kalman pour décomposer un tracé manuscrit en lignes droites et en arcs de cercle. Ces primitives sont bien adaptées à l'interprétation de plans architecturaux.

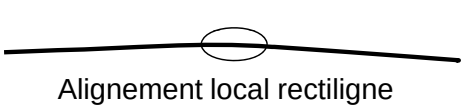
Nous adoptons dans notre approche IMISketch la méthode présentée dans [102] basée sur le filtre de Kalman pour l'extraction des segments de droite. Le filtre de Kalman

est un processus de vérification/prédiction qui fournit une estimation d'un modèle à partir d'observations. Cette méthode était déjà présente dans l'équipe *IntuiDoc* et ne constitue pas le cœur de notre travail. Nous la présentons rapidement ici. Dans le cas de l'extraction des segments, le modèle est basé sur trois valeurs : l'épaisseur, la position et la pente locale de la droite. L'un des points forts de ce système est de permettre le suivi des segments dans un environnement bruité. En effet, il gère le biais, l'intersection entre plusieurs segments et la traversé des segments discontinus. Lorsque le dessin est relativement régulier, le filtre de Kalman peut utiliser ces trois valeurs pour prédire la position suivante, et mettre à jour le modèle à partir des pixels de l'image. Dans les zones contenant des intersections, l'épaisseur observée n'est pas compatible avec l'épaisseur prédite, mais il est possible d'utiliser la capacité de prédiction du filtre afin de passer à travers ces régions. Dans une région où la direction de la ligne change rapidement, il n'est pas possible de trouver des pixels noirs dans la position prévue et le suivi des segments s'interrompt. Les équations et plus de détails peuvent être trouvés dans [102]. La figure 4.2 illustre des exemples d'interprétation réalisées par le filtre de Kalman. La méthode adaptée permet de faire face :

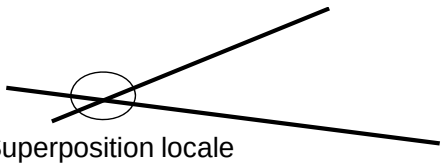
- à l'existence possible de discontinuités : il est utile de permettre localement une absence de points, due à la qualité ou à la nature des objets extraits (ligne pointillée, défaut de binarisation, bruit) (figure 4.2(a)) ;
- prise en compte de l'épaisseur pour chaque point représentatif (figure 4.2(b)) ;
- prise en compte des segments qui se croisent (figure 4.2(c)) ;
- prise en compte de la courbure dans un segment (figure 4.2(d)) ;

(a) Deux segments, dont on intègre les discontinuités

(b) Un seul segment fin, recouvert partiellement par un second segment plus épais



(c) Un segment globalement courbe

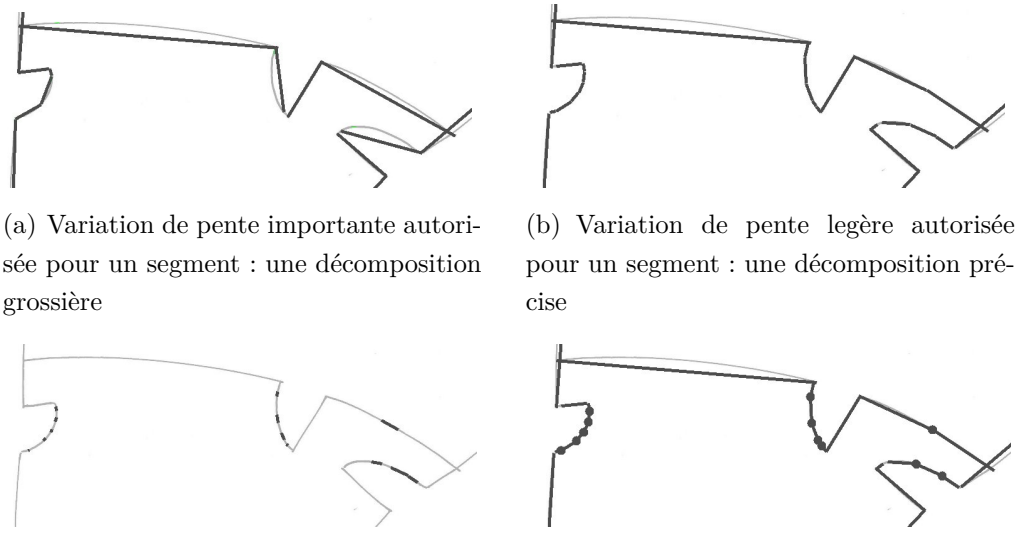


(d) Deux segments qui se croisent

Figure 4.2 – Exemples de détections réalisées par l'extracteur de segments utilisé

L'utilisation du filtrage de Kalman nécessite de régler quelques paramètres. Pour les documents structurés dessinés à main levée, le paramètre le plus important du filtre de Kalman est celui qui définit la variabilité de la pente. Avec une valeur élevée (0.1,c'est-à-dire une variation de la pente de 10%), chaque courbe du document est

détectée comme un seul segment joignant ses deux extrémités (figure 4.13(a)). Avec une valeur très faible (0,005), le modèle permet une variation très petite de la pente, et donc de nombreux petits segments sont détectés pour la même courbe (figure 4.13(b)). Notons que, dans cet exemple, les symboles représentant les ouvrants (porte ou fenêtre) sont plus précisément représentés dans la figure 4.13(b) (décomposition précise) que dans la figure 4.13(a) (décomposition grossière). Une décomposition précise augmente le nombre de segments extraits de l'image, mais cette décomposition peut être très utile lorsque nous utilisons un classifieur pour la reconnaissance de symboles.



(a) Variation de pente importante autorisée pour un segment : une décomposition grossière

(b) Variation de pente legère autorisée pour un segment : une décomposition précise

(c) En noir la partie commune de deux segments

(d) Représentation des polygones, les cercles représentent les connexions entre les segments d'un même polygone

Figure 4.3 – Extraction de primitives. L'image originale est en gris clair, et les primitives extraites sont en noir.

Pour fixer les idées, le plan complet, dont la figure 4.3 est extraite, est représenté par 144 segments pour la décomposition grossière et 177 segments pour la décomposition précise. Cette augmentation du nombre de segments est principalement due à de petits segments dans les régions des ouvrants. Il faut remarquer qu'une région réduite contenant de nombreux petits segments peut engendrer une explosion combinatoire dans les phases d'analyse ultérieures.

Pour surmonter ce dilemme, c'est-à-dire une représentation précise tout en limitant la combinatoire, nous proposons d'extraire une décomposition précise en établissant des relations spatiales entre les segments appartenant à des zones très courbes. Lors de la détection de segments par le filtre de Kalman, il est possible de détecter qu'une

partie du dessin appartient à deux segments différents (figure 4.3(c)). Dans ce cas, nous enlevons cette partie commune de l'un des deux segments et nous mémorisons qu'il y a un lien linéaire entre les extrémités de ces deux segments. Les petits cercles dans la figure 4.3(d) représentent une connexion entre deux segments. Chaque ensemble de segments attachés entre eux correspond à une approximation polygonale des parties courbes du dessin (par exemple un ouvrant).

Il existe alors deux types de primitives extraites d'une image : les segments et les polygones. Nous mémorisons les informations de connexion de telle sorte que le processus d'interprétation peut utiliser directement les segments et les polygones en tant que primitive, ou peut également décider de diviser un polygone en segments selon certaines informations structurelles telle que la taille du polygone et la colinéarité. L'utilisation de cette double représentation permet, d'une part, une présentation plus précise des segments de courbe et de limiter, d'autre part, la combinatoire au cours du processus d'interprétation en travaillant au niveau de la primitive polygone. Nous approfondirons dans les sections suivantes comment ces primitives sont utilisées conjointement et quel impact de cela sur la réduction de l'explosion combinatoire.

4.2 Modélisation des connaissances a priori

IMISketch peut être caractérisé par l'exploitation de deux natures de connaissance : structurelles et statistiques. Ces connaissances peuvent être introduites séparément pour chaque type de document à reconnaître, sans avoir besoin de modifier l'analyseur. Les connaissances structurelles sont modélisées par les grammaires GMC-PC. L'objectif des connaissances structurelles est de piloter l'analyseur dans la structure bidimensionnelle du document. Les connaissances statistiques sont formalisées par des classifieurs. Les connaissances structurelles vont aussi permettre de cibler les appels aux classifieurs en réduisant les classes en concurrence en fonction du contexte qui aura été identifié. La fusion de ces deux typologies de connaissance complémentaires (structurelle et statistiques) va permettre de mettre en place un processus de décision robuste qui sera au cœur d'IMISketch interactif. L'ensemble de ces notions va être développé dans les sections suivantes.

4.2.1 Grammaires GMC-PC

La description structurelle consiste à décrire l'agencement spatial entre les constituants du document. Le choix du formalisme grammatical a été établi dans notre approche pour maintenir une cohérence des formalismes dans le continuum, c'est-à-dire

une cohérence entre la description utilisée pendant la rétroconversion et la description utilisée pendant l'étape de la composition/édition.

Nous avons choisi les grammaires GMC-PC, conçues dans l'équipe IntuiDoc [108], pour l'interprétation en-ligne à la volée des documents bidimensionnels. Ce formalisme grammatical n'a pas été conçu ni pour la rétroconversion, ni pour l'interprétation hors-ligne. Par conséquent, des adaptations et des améliorations sont nécessaires pour adapter ce formalisme à la reconnaissance hors-ligne ou a posteriori. Ces améliorations portent essentiellement sur la résolution des problèmes d'explosion de la combinatoire. En effet, dans l'interprétation à la volée, l'analyseur interprète une primitive (tracé) dans un document déjà interprété. En rétroconversion, nous avons plusieurs primitives à interpréter dans un contexte de document partiellement reconnu. Toutes les primitives peuvent être interprétées dans tous les sens ce qui engendre une très grande combinatoire qu'il va falloir maîtriser.

La première adaptation proposée pour les GMC-PC est l'introduction d'un paramètre précisant le type d'exploration. En effet, les arbres d'analyse sont construits à base des règles de production GMC-PC. L'exploration hybride des arbres d'analyse, présentée dans la section 4.3.2.2.b, est pilotée par les grammaires GMC-PC, ce qui est une originalité introduite dans cette thèse. Nous avons donc introduit pour chaque règle de production une nouvelle étiquette nommée *«Type d'exploration»*, qui prend comme valeur 'largeur' ou 'profondeur'. Une règle de production ayant un type d'exploration 'profondeur' est une règle appliquée en profondeur, c'est-à-dire une règle qui ne sera pas mise en concurrence avec d'autres interprétations. Une règle étiquetée 'largeur' est une règle qui peut être mise en concurrence avec d'autres interprétations. La figure 4.4 présente la nouvelle forme de la règle de production. Cet exemple est enrichi à partir de l'exemple introduit dans la figure 3.1.

Le second point fort de notre adaptation se situe au niveau des terminaux. Pour l'interprétation en-ligne les GMC-PC présentent un seul terminal qui est le tracé, tandis que, pour la rétroconversion, nous considérons deux terminaux : les segments et les polygones, déduits de la phase d'extraction des primitives décrite dans la section 4.1.

Nous avons donc enrichi dans cette partie les grammaires GMC-PC pour qu'elle soit adaptée à la fois à la rétroconversion et au signal hors-ligne des documents. Ce formalisme enrichi permet de piloter l'exploration des arbres d'analyse et de choisir entre l'exploration en largeur ou en profondeur, tout en gérant plusieurs types de primitives.

Mur : mRes $\rightarrow$ **Segment t : t** avec { **Type d'exploration :**$'largeur'$

Préconditions :

$\{(Mur : m1)[extrémité]t[un]\}$

Contraintes :

traceLong(t) &
reconnaisseurSegment(t, Mur)

Postconditions :

$\{mRes[extrémité](Segment : tn1)[un] \Rightarrow [Mur \rightarrow tn1]\}^*$ &
$\{mRes[milieu](Segment : tn2)[tous] \Rightarrow [Porte \rightarrow tn2]\}^*$

Figure 4.4 – Production GMC-PC pour la composition d'un mur avec type d'exploration

4.2.2 Classifieur

Dans les documents structurés, tels que les plans d'architecture, on trouve de très nombreux symboles qui, de plus, peuvent être représentés de façons différentes quand ils sont tracés à la main dans des croquis. Pour faire face à cette variabilité, nous allons exploiter un système de reconnaissance de symboles auto-évolutif capable d'ajouter de nouveaux symboles durant l'analyse. Almaksour [8] propose un système d'apprentissage incrémental de classifieurs qui peut commencer à apprendre à partir de zéro et avec peu de données d'apprentissage. Il peut aussi s'améliorer et s'adapter aux données nouvellement disponibles.

Ce système de classification est basé sur les systèmes d'inférence floue SIF de type Takagi-Sugeno d'ordre 1 [154]. Ce système de classification à base de prototypes est flexible et capable de s'adapter et d'évoluer selon les nouvelles données.

Les règles floues font le lien entre les modèles intrinsèques (prémisses) et les sorties du système par des fonctions «conséquences». Pour un problème à K classes, une règle R_i est construite pour chaque modèle floue P_i :

$$\textbf{Règle}_i : \textbf{SI } \vec{x} \textit{ est } P_i \textbf{ ALORS } y_i^1 = l_i^1(\vec{x}), ..., \ y_i^k = l_i^k(\vec{x}) \tag{4.1}$$

avec k est le nombre de classes et $l_i^m(\vec{x})$ représente les fonctions conséquences linéaires de la règle i pour la classe m :

$$l_i^m(\vec{x}) = \vec{\pi}_i^m \vec{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \tag{4.2}$$

avec n la taille du vecteur d'entrée. $a_i^m j$ est un coefficient dans la règle i entre le score de la classe m et le descripteur j du vecteur d'entrée. Le prototype P est définie par un centre et une zone d'influence floue. Pour trouver la classe de x, son degré

d'appartenance à chaque prototype floue $\beta_i(x)$ est calculé. Après la normalisation de ces degrés, l'inférence floue de type somme-produit est ensuite utilisée pour calculer la sortie du système pour chaque classe :

$$y^m(\vec{x}) = \sum_{i=1}^{r} \bar{\beta}_i(\vec{x})\ l_i^m(\vec{x}) \tag{4.3}$$

avec r est le nombre de règles dans le système. Le score de chaque classe y^m est entre 0 et 1. L'étiquette de la classe gagnante est donnée en trouvant la sortie maximale et en prenant l'étiquette de classe correspondante comme réponse :

$$class(\vec{x}) = y = argmax\ y^m(\vec{x}) \qquad m = 1, .., k \tag{4.4}$$

Les degrés sont calculés à partir du centre du prototype μ_i et la matrice de covariance A_i en utilisant la probabilité de Cauchy :

$$\beta_i(\vec{x}) = \frac{1}{2\pi\sqrt{|A_i|}} \left[1 + (\vec{x} - \vec{\mu_i})^t A_i^{-1} (\vec{x} - \vec{\mu_i})\right]^{-\frac{n+1}{2}} \tag{4.5}$$

L'algorithme d'apprentissage incrémental se compose de trois tâches différentes : la création de nouvelles règles, l'adaptation des prémisses des règles existantes, et le réglage des paramètres linéaires conséquentes. Ces trois tâches doivent être effectuées dans un mode incrémental et tous les calculs nécessaires doivent être complètement récursive.

Nous avons couplé IMISketch avec ce type de classifieur, ce qui permet de gérer l'introduction de connaissances sur les formes de symboles à reconnaitre.

4.3 Construction des arbres d'analyse

La phase de construction des arbres d'analyse consiste à chercher les hypothèses possibles pour interpréter un élément du document. Pour réduire l'espace de recherche des interprétations, nous limitons l'exploration du contexte pour l'interprétation d'une primitive à une zone du document appelée contexte local de recherche.

4.3.1 Définition du contexte local de recherche

La reconnaissance d'une primitive donnée dépend de son voisinage dans les documents structurés. Notre analyseur commence par la définition d'un contexte local de recherche qui vise à limiter l'exploration des différentes hypothèses possibles permettant l'interprétation de la primitive. Vu que nous travaillons sur les documents structurés

bidimensionnels, nous adoptons un contexte local de recherche bidimensionnel. Vis à vis de l'arbre d'analyse, cette zone bidimensionnelle locale est définie comme la distance maximale entre les éléments de la racine et les éléments de toutes les feuilles.

Le choix de la taille du contexte local dépend du domaine d'application. Par exemple, pour interpréter un plan d'architecture, nous suggérons de prendre un contexte local avec une taille correspondant à la taille maximale d'un ouvrant (porte, fenêtre, etc.).

Les figures 4.5 et 4.6 illustrent le déplacement du contexte local entre deux étapes d'analyse consécutives. Après chaque étape, le contexte local de recherche se déplace et il sera piloté par le prochain élément à analyser. Ce déplacement permet de voir l'impact de la reconnaissance d'un élément sur son voisinage.

Une fois le contexte local défini, l'analyseur passe à la construction des arbres d'analyse. Seuls les éléments présents dans le contexte local seront pris en compte pour la construction des arbres d'analyse.

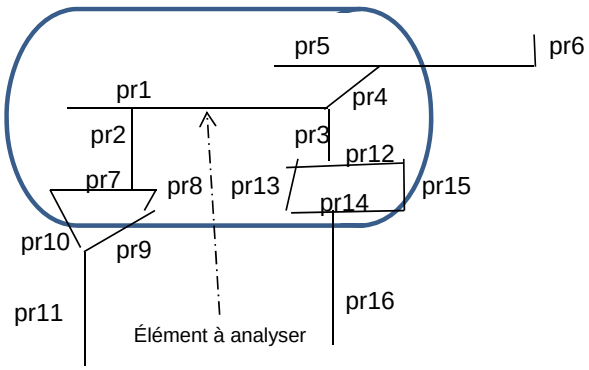


Figure 4.5 – Le contexte local à l'étape *s* pour l'analyse de l'élément *pr1*

4.3.2 Construction des arbres d'analyse

Dans cette étape, l'analyseur explore toutes les hypothèses possibles d'interprétation dans le contexte local de recherche en utilisant un ensemble de règles de productions décrivant la structure du document. La stratégie d'exploration pour avoir un ensemble d'hypothèses concurrentes exige de développer une exploration en largeur des arbres d'analyse. L'extraction d'hypothèses concurrentes pour interpréter une primitive va permettre de nourrir le processus de décision qui pourra ainsi soit décider d'interpréter automatiquement la primitive soit de solliciter l'utilisateur. Nous présentons au début de cette section, l'exploration en largeur des arbres d'analyse, nous montrons ensuite

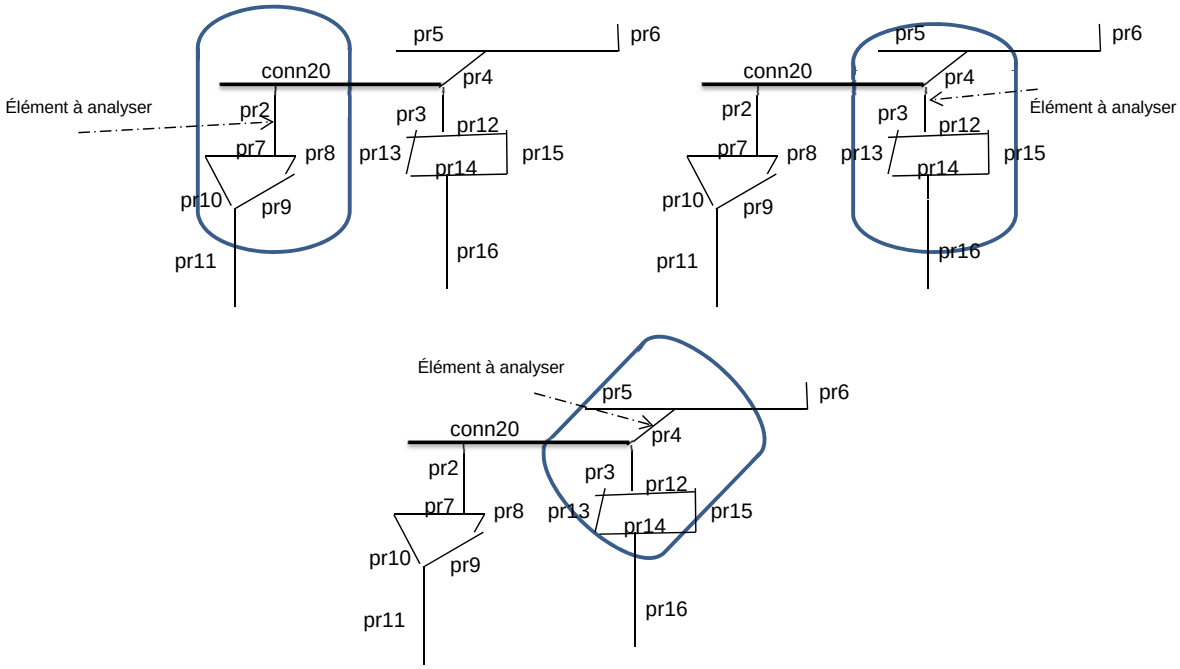


Figure 4.6 – Le contexte local à l'étape s+1

les optimisations introduites pour réduire la combinatoire engendrée. Les optimisations portent sur des contraintes structurelles ajoutées à l'exploration en largeur et sur une exploration hybride alternant l'exploration en largeur et en profondeur. Nous détaillons dans la suite de cette section la méthode classique *IMISketch classique* basée sur une exploration en largeur, ensuite les optimisions structurelles, dans la méthode nommée *IMISketch optimisé* enfin la construction hybride des arbres d'analyse, dans la méthode nommée *IMISketch hybride*. Notons que ces optimisations visent à éviter la construction des hypothèses inutiles.

4.3.2.1 Construction en largeur des arbres d'analyse

L'exploration en largeur consiste à explorer toutes les hypothèses possibles d'interprétation dans la limite du contexte local de recherche. Chaque primitive peut être interprétée de plusieurs manières. Chaque nœud ou feuille correspond à une règle de production déduite à partir du nœud parent. Chaque feuille ou nœud de l'arbre a un score calculé à partir de son score local et du score obtenu à partir des nœuds parent. Chaque score détermine le degré d'adéquation d'une production. Le score de production peut également être déduit à partir d'un classifieur. Par conséquent, chaque branche (hypothèse) est caractérisée par un score. Nous reviendrons dans la section 4.4 dédiée au processus de décision sur le détail du calcul de ce «scoring». Un exemple d'exploration en largeur en utilisant le contexte local de recherche est illustré dans les figures 4.9

et 4.10. Cette exploration peut engendrer de l'explosion combinatoire. Par conséquent, nous décrivons dans la section 4.3.2.2 des optimisations au niveau de la construction des arbres d'analyse pour réduire cette combinatoire.

Les figures 4.7 et 4.8 montrent un exemple d'exploration en largeur sur l'ensemble de primitives présenté dans les figures 4.5 et 4.6 en appliquant les règles de production illustrées dans le tableau 4.1. Dans cet exemple nous avons trois types de symboles : *connexion*, *triangle* et *rectangle*. Un triangle et un rectangle sont constitués d'un ensemble de primitives entre deux connexions colinéaires. Chaque primitive peut être une connexion.

En appliquant les règles du tableau 4.1 sur l'élément 'pr1', nous constatons que cet élément ne peut être interprété qu'en un connexion. L'interprétation de cette primitive engendre des hypothèses d'interprétation possibles pour les éléments voisins (contenus dans le contexte local de recherche). L'ensemble des ces hypothèses est présenté dans la figure 4.7. En passant à l'étape suivante *(s+1)*, le contexte local de recherche se positionne sur les éléments consommés dans les fils directs de l'élément précédemment analysé. Ce déplacement permettra de continuer la construction des arbres en présence des nouveaux éléments contenus dans le contexte local et non de reconstruire les arbres. Les nœud grisés dans la figure 4.8 présentent les nœuds nouvellement ajoutés dans les arbres d'analyse.

Règle	**Éléments créés**	**Éléments consommés**
P_{conn}	Connexion	**Primitive** du document
P_{rect}	Rectangle	Ensemble de **primitives** entre deux connexions colinéaires
P_{trian}	Triangle	Ensemble de **primitives** entre deux connexions colinéaires

Tableau 4.1 – Règles de production appliquées aux exemples 4.5 et 4.6

4.3.2.2 Optimisation de la construction des arbres d'analyse

L'exploration en largeur des arbres d'analyse dans une méthode de rétroconversion peut engendrer de l'explosion combinatoire. Nous proposons de mettre en place

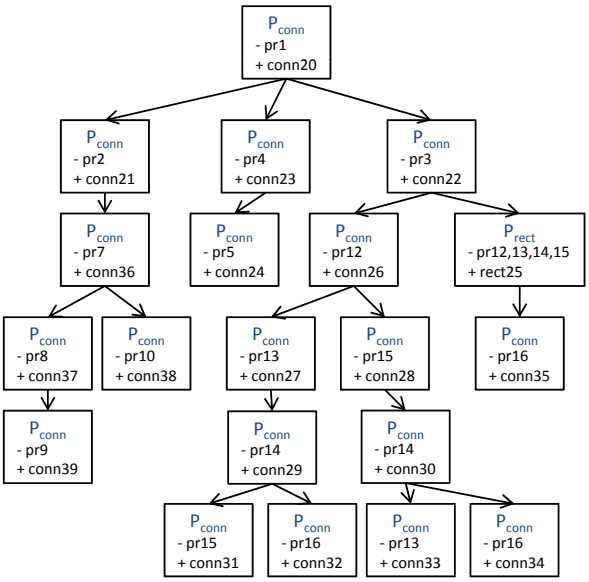


Figure 4.7 – Arbre d'analyse associé à la primitive 'pr1' de l'exemple illustré dans la figure 4.5 en appliquant les règles présentées dans le tableau 4.1

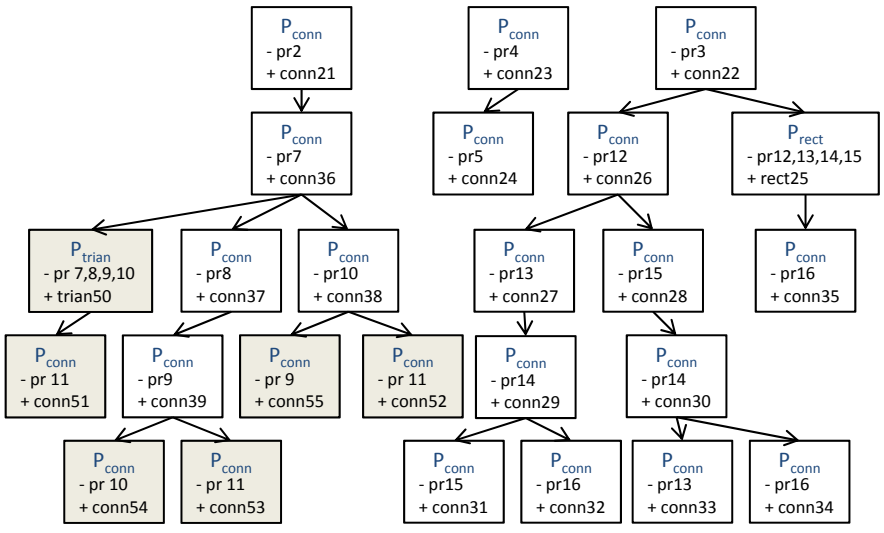


Figure 4.8 – Arbres d'analyse associés à aux primitives 'pr2', 'pr3' et 'pr4' de l'exemple illustré dans la figure 4.6 en appliquant les règles présentées dans le tableau 4.1. Les nœuds grisés correspondent aux nœuds nouvellement ajoutés en se déplaçant le contexte local de recherche.

plusieurs stratégies d'optimisation en se basant notamment sur la présence de l'utilisateur durant l'analyse pour réduire la combinatoire. Une première stratégie se base sur l'exploitation de contraintes structurelles. Une seconde stratégie cherche à mettre en place une exploration hybride reposant sur une alternance entre l'exploration en largeur et l'exploration en profondeur au sein d'un arbre d'analyse afin de ne garder que les hypothèses qui sont vraiment en concurrence.

4.3.2.2.a Contraintes structurelles pour l'exploration en largeur

Chaque arbre d'analyse caractérise les éléments à interpréter dans le contexte local de recherche déjà défini. Chaque racine est une règle de production consommant cette primitive. Le nombre d'arbres d'analyse correspond au nombre d'interprétations possibles de la primitive courante. L'arbre d'analyse contient alors un ensemble d'objets *complets* ou *incomplets*. Un objet est dit *complet* si et seulement si cet objet intervient dans le résultat final de l'interprétation du document. Par exemple, dans le cas des plans d'architecture, les objets complets sont les murs, les portes, les fenêtres, etc. Un objet *incomplet* est un objet intermédiaire qui ne peut pas être représenté dans le résultat final de l'analyse. Dans le cas de l'analyse du plan d'architecture, il existe des multiples objets incomplets tels que l'ensemble de segments représentant le début d'un ouvrant.

Malgré l'intégration du contexte local de recherche, le problème de combinatoire, et donc de temps de calcul, reste une limite forte pour répondre aux critères d'acceptabilité d'un tel système d'analyse interactive avec un utilisateur final. La combinatoire engendrée est principalement liée au nombre de règles de production appliquées dans le contexte local de recherche.

En fait, nous constatons que l'exploration de certains nœuds est inutile, c'est-à-dire qu'elle n'apporte pas vraiment une information supplémentaire sur le choix de la bonne interprétation. Dans l'exemple de la figure 4.5, nous remarquons qu'il n'y a qu'une seule façon d'interpréter la primitive 'pr1'. Dans ce cas, le système d'analyse n'a pas besoin de prendre en compte les interprétations des éléments voisins de la primitive 'pr1' pour interpréter 'pr1' et la transformer en une connexion.

De plus, dans le cas des documents structurés, l'ordre d'analyse d'un élément n'influe pas sur son interprétation. Par conséquent, nous pouvons intervenir sur l'ordre d'analyse des éléments de la racine d'une manière intelligente, sans avoir d'impact sur le résultat de l'interprétation du document.

Pour cela, nous présentons une nouvelle méthode (*IMISketch optimisé*) basée sur une optimisation de l'exploration en largeur à travers des contraintes structurelles, afin

de ne développer que les nœuds utiles pour prendre la bonne décision. Ces optimisations sont introduites dans un algorithme décrit ci-dessous :

– Si le nombre d'arbres d'analyse est égal à 1 :

  limiter l'exploration aux fils directs de la racine. Dans le cas où il n'y a qu'une seule racine à explorer, nous pouvons dire qu'une seule interprétation est possible pour transformer la primitive courante. Le processus de décision connait d'avance l'interprétation à valider. Il est donc inutile d'explorer tout l'arbre d'analyse et nous pouvons limiter la construction de l'arbre uniquement aux fils directs. Une fois la racine validée, ses fils directs seront les nouvelles racines des arbres à construire.

– Si le nombre d'arbres d'analyse est supérieur à 1 :

  1. regrouper les arbres selon les éléments consommés à la racine. Un groupe d'arbre est l'ensemble des arbres qui analysent les mêmes primitives à leurs racines. L'analyseur ne construit pas tous les arbres mais seulement les arbres appartenant au même groupe. Les racines des arbres du même groupe indiquent les différentes manières pour interpréter la primitive à la racine. Au lieu de mettre en compétition toutes les hypothèses, l'analyseur met en concurrence uniquement les hypothèses d'interprétations de la primitive courante.
  2. ordonner ces groupes selon le nombre d'arbre qu'ils contiennent.
  3. construire seulement les racines du groupe ayant le moins d'arbres. Nous considérons que construire le minimum d'arbre permet de gagner plus en temps de calcul. La construction est lancée tant que les conditions suivantes sont satisfaites :

     . le dernier élément consommé est dans le contexte local de la recherche (condition utilisée également par IMISketch classique) ;

     . le nombre d'éléments consommés dans chaque hypothèse (branche) est inférieur à un seuil $S_{primitives}$ déterminé a priori. Ce seuil dépend de la complexité des documents à analyser. Dans les plans architecturaux, nous fixons ce seuil à 10, ce qui correspond au nombre maximal de primitives d'un ouvrant.

     . le nombre d'éléments *complets* d'une branche est inférieur à un seuil ($S_{complet}$). Nous prenons $S_{complet} = 3$ pour les plans architecturaux.

Les figures 4.11 et 4.12 illustrent un exemple de la construction des arbres d'analyse avec notre nouvelle méthode (*IMISketch optimisé*). Le contexte local ne se limite pas uniquement à la distance entre les primitives, mais aussi au nombre d'éléments

complets dans chaque hypothèse. Cette optimisation peut générer dans certains cas un manque d'information sur des hypothèses et des ambiguïtés peuvent donc persister. Mais grâce à l'interactivité, cela a peu d'impact sur le résultat final dans la mesure où l'utilisateur peut être sollicité pour valider la bonne hypothèse et donc pallier ce manque d'information.

Nous allons présenter dans la section suivante une comparaison entre *IMISketch classique* et *IMISketch optimisé* afin de montrer les avantages et les améliorations liées à *IMISketch optimisé* en détaillant un exemple d'analyse.

IMISketch classique Vs IMISketch optimisé

Le but de cette partie est de montrer les améliorations apportées dans ce nouvel algorithme (IMISketch optimisé) pour la construction des arbres d'analyse par rapport à la méthode d'exploration IMISketch classique (dans lequel toutes les branches possibles sont explorées dans le contexte local). Pour faciliter cette comparaison, nous présentons un exemple d'arbre artificiel d'analyse. L'objectif est de comparer, pour chaque étape, le nombre de nœuds construits en fonction de la mise en place du contexte local par rapport aux éléments à analyser. La transition d'un arbre d'analyse à l'arbre suivant engendre un déplacement du contexte local. Ce déplacement permet d'utiliser un nouveau contexte, centré sur la nouvelle primitive à analyser, pour appliquer d'autres productions. Il est important de remarquer ici que, la construction d'un arbre d'analyse n'engendre pas la reconnaissance d'une branche en entier, mais uniquement l'interprétation de la racine avec sa production associée.

La figure 4.9 présente un exemple générique de résultat de l'exploration en utilisant *IMISketch classique*. Le nombre de nœuds est égal à 80. La construction de l'arbre en utilisant *IMISketch optimisé* génère seulement 6 interprétations (figure 4.11). En effet, comme il n'y a qu'une seule manière d'interpréter la racine, on peut se contenter d'étudier les fils directs. Les figures 4.10 et 4.12 illustrent les nouveaux arbres d'analyse en déplaçant le contexte local. Sur deux étapes successives de construction des arbres d'analyse pour les deux méthodes (*IMISketch classique* et *IMISketch optimisé*), nous constatons qui nous sommes passés de 111 interprétations (*IMISketch classique*) à 28 interprétations (*IMISketch optimisé*), ce qui présente un gain d'environs 75% du temps de calcul. Ces optimisations sont génériques et ne dépendent pas de la catégorie de document structurés à analyser.

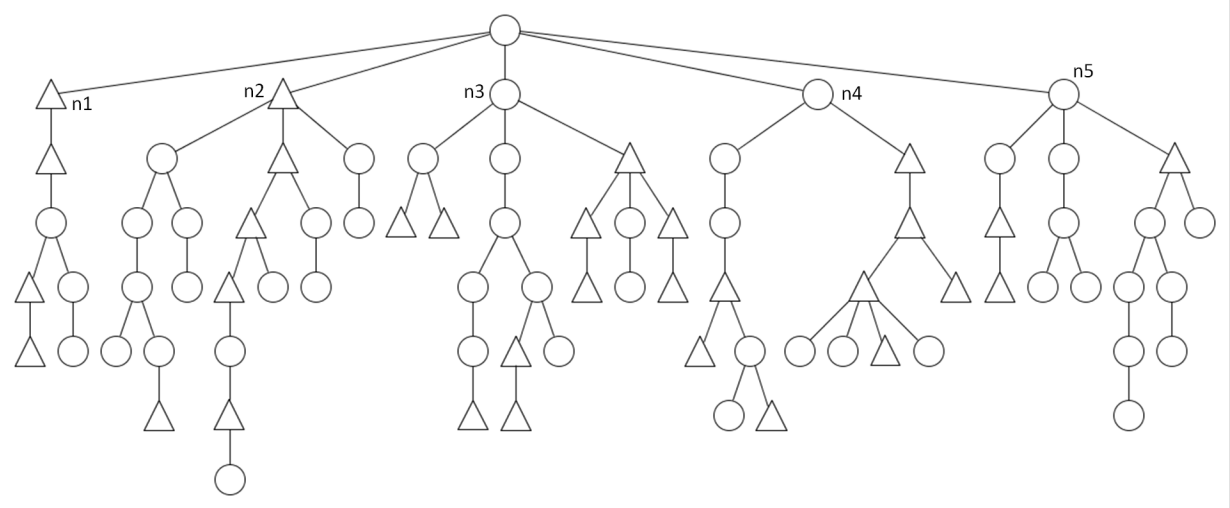


Figure 4.9 – Méthode de construction de l'arbre d'analyse de **IMISketch classique**. Le nombre de nœuds nouvellement créés = 80

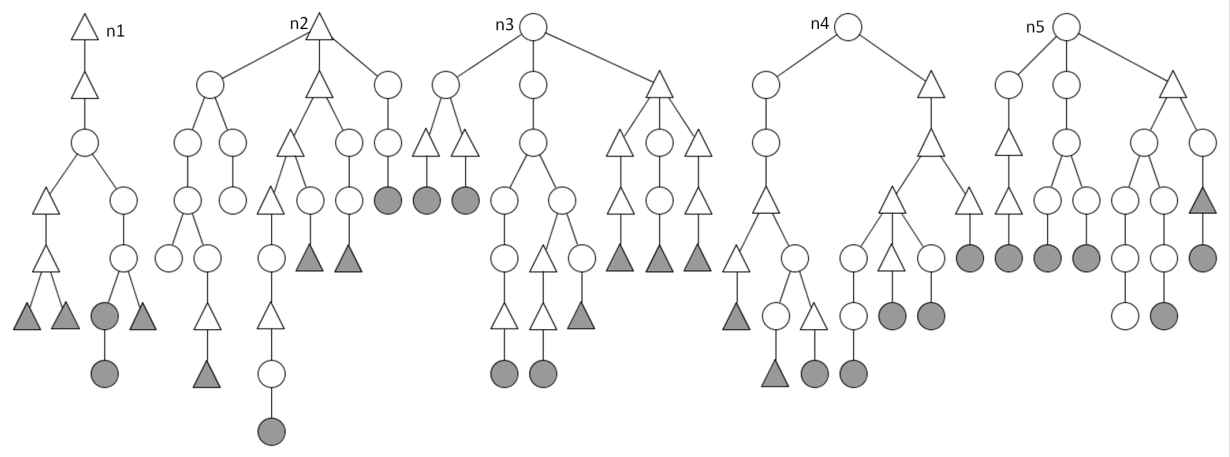


Figure 4.10 – Méthode de construction de l'arbre d'analyse de **IMISketch classique**. Les nœuds modélisés par des triangles désignent des objets incomplets. Les nœuds grisés correspondent aux nouvelles règles de production appliquées en déplaçant le contexte local de recherche.

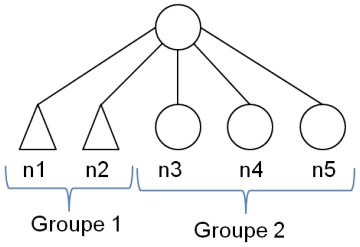


Figure 4.11 – Méthode de construction de l'arbre d'analyse de **IMISketch optimisé**. Le nombre de nœuds nouvellement créés = 6. Les fils directs de la racine sont groupés par éléments consommés, pour former deux groupes.

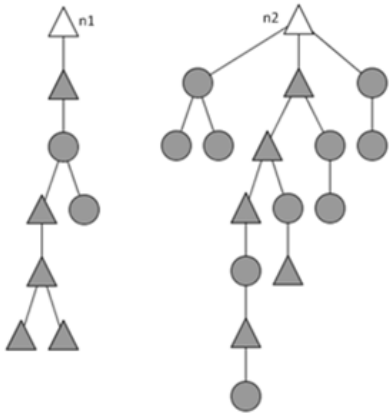
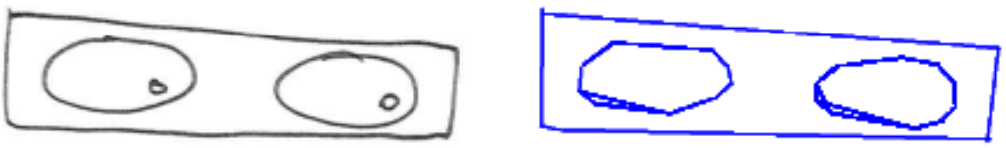


Figure 4.12 – Méthode de construction de l'arbre d'analyse de **IMISketch optimisé**. Les racines n1 et n2 développées appartiennent au même groupe (Groupe 1) (elles partagent les mêmes éléments consommés). Le nombre de nœuds nouvellement créés = 22. Les nœuds modélisés par des triangles désignent des objets incomplets. Les nœuds grisés correspondent aux nouvelles règles de production appliquées en déplaçant le contexte local de recherche.

4.3.2.2.b Construction hybride des arbres d'analyse

Les symboles complexes, comme les mobiliers dans le cas des plans d'architecture (figure 4.13), génèrent souvent de l'explosion combinatoire, car ils sont composés d'un grand nombre de primitives interconnectées dans une zone réduite. L'avantage de l'exploration en largeur est d'offrir la possibilité de choisir entre plusieurs hypothèses possibles pour l'interprétation des primitives. Cependant, cette exploration génère de l'explosion combinatoire surtout quand il s'agit d'une condensation forte des primitives dans des zones réduites. Réciproquement, l'exploration en profondeur limite les problèmes de combinatoires mais malheureusement ne génère pas toutes les hypothèses possibles.



(a) Exemple de mobilier : un lavabo (b) Primitives extraites

Figure 4.13 – Exemple de mobilier et les primitives associées

Dans un document structuré, chaque symbole est caractérisé par sa forme graphique et le contexte structurel où il doit être localisé. Dans un plan d'architecture, un mur est graphiquement un segment et structurellement une primitive appartenant au document. Un ouvrant correspond graphiquement à un ouvrant (porte, fenêtre, fenêtre coulissante, etc.) et structurellement un ouvrant est un ensemble de primitives entre deux murs colinéaires. Un mobilier correspond graphiquement à une forme de mobilier (chaise, lit, table, canapé, etc) et structurellement un ensemble de primitives très proches dans une zone réduite et à l'intérieur de murs. Le tableau 4.2 illustre quelques exemples de symboles des plans d'architecture. Pour faciliter la compréhension, nous illustrons l'exemple d'un ouvrant sur une fenêtre et l'exemple du mobilier sur un évier.

Symbole	Forme graphique	Contexte structurel
Mur		Fait partie du plan
Ouvrant		Entre deux murs colinéaires
Mobilier		Des primitives très proches dans une zone réduite et à l'intérieur d'un plan

Tableau 4.2 – Caractéristiques des symboles du plan d'architecture

Un symbole S est bien reconnu si seulement si les deux critères sont vérifiés. Cette approche est particulièrement intéressante car elle consiste à ne passer à la reconnaissance de la forme graphique qu'une fois la reconnaissance du contexte structurel du symbole bien satisfaite (réussie). La stratégie proposée consiste à satisfaire le plus rapidement possible le contexte structurel. Pour cela, nous adoptons l'exploration en profondeur tant que le contexte structurel n'est pas satisfait. Si le contexte structurel est satisfait, nous considérons que deux possibilités sont possibles : soit l'analyseur reconnait l'objet obtenu graphiquement en se basant sur un classifieur, soit il continue la recherche du prochain contexte structurel vérifié. Cette stratégie est ainsi hybride puisqu'elle combine les deux types d'exploration : en largeur et en profondeur.

Dans la section suivante, nous allons comparer *IMISketch optimisé* à *IMISketch hybride* pour mettre en valeur le passage à une approche hybride en détaillant un exemple d'interprétation.

IMISketch optimisé Vs IMISketch hybride

Dans cette partie, nous allons montrer l'intérêt de l'utilisation d'une approche hybride pour d'une part garantir la maintenance des hypothèses concurrentes et d'autres part éviter d'explorer les hypothèses qui n'interviennent pas au niveau de la prise de décision. Nous présentons cette amélioration à travers un exemple de plan d'architecture. Nous voulons reconnaitre l'ensemble de primitives présentées dans la figure 4.14. Le tableau 4.3 présente les règles de production utilisées pour l'interprétation du plan. Notons que dans cet exemple, nous considérons deux objets incomplets (objets intermédiaires) : *séquence ouvrant* et *séquence mobilier* et trois objets complets : *mur*, *ouvrant*, *mobilier*. Pour montrer le gain introduit par la construction hybride (largeur/profondeur), nous allons comparer les arbres d'analyse construits en utilisant *IMSISketch optimisé* avec ceux d'*IMISketch hybride*. Nous lançons les deux versions (*IMSISketch optimisé* et *IMISketch hybride*) sur l'ensemble de primitives illustrées dans la figure 4.14 en respectant la description décrite par les règles de production présentées dans le tableau 4.3.

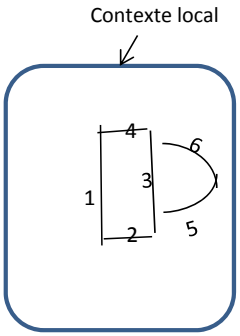


Figure 4.14 – Les primitives à interpréter

En fait, la construction optimisée est basée sur une exploration en largeur permettant à chaque étape de reconnaitre un symbole structurellement et graphiquement simultanément. À chaque nœud, l'analyseur reconnait le symbole obtenu graphiquement même s'il n'est pas bien reconnu structurellement. Ceci engendre des hypothèses d'interprétation des symboles qui ne vérifient pas à la fois le contexte structurel du symbole et sa forme graphique.

En appliquant les règles de production (tableau 4.3) sur les primitives de la figure 4.14, nous constatons que deux interprétations sont possibles pour la primitive '1' : un mur ou une partie d'un mobilier. Par conséquent, deux arbres d'analyse seront générés. Le nombre d'arbres construits est identique dans les différents versions IMISketch.

Règle	Éléments créés	Éléments consommés	Type d'exploration
Ps	Mur départ	Primitive le plus longue dans un document	Largeur
P1	Mur	Primitive à l'extrémité d'un mur	Largeur
P2	Séquence ouvrant	Primitive à l'extrémité d'un mur ou à l'extrémité d'une séquence ouvrant	Largeur
P3	Séquence mobilier	Primitive à l'extrémité d'un mur	Largeur
		primitive à l'extrémité **étendue** d'une séquence mobilier	Largeur
		primitive à l'extrémité **réduite** d'une séquence mobilier	Profondeur
P4	Ouvrant	Séquence ouvrant entre deux mur colinéaires ou un mur et une primitive colinéaire et semblable à un ouvrant	Largeur
P5	Mobilier	Séquence mobilier semblable à un mobilier	Largeur

Tableau 4.3 – Description des plans d'architecture. Uniquement la règle, qui transforme une primitive à l'extrémité d'une *séquence mobilier* en une *séquence mobilier*, est appliquée en *profondeur*. Toutes les autres règles sont appliquées en largeur.

Le premier arbre a pour but de chercher toutes les hypothèses générées dans le contexte local de recherche si la primitive '1' est un mur, alors que le deuxième arbre permet de chercher l'impact de l'interprétation de la primitive '1' en une partie du mobilier sur ces voisins.

L'arbre, illustré par la figure 4.15, qui interprète la primitive '1' en un mur est identique entre *IMISketch optimisé* et *IMISketch hybride*. D'après la règle de production, il suffit que la primitive soit dans le document pour satisfaire le contexte local d'un mur, la construction des murs est toujours en largeur. En fait, vu que le contexte structurel d'un mur est toujours vérifié, nous pouvons déduire que l'application de la règle de production qui transforme une primitive en un mur engendre une exploration en largeur.

La figure 4.16 illustre l'arbre d'analyse pour la transformation de la primitive '1' en un mobilier en utilisant *IMISketch optimisé*. Selon les règles de production définies dans le tableau 4.3, la transformation de la primitive '1' en un mobilier avec *IMISketch optimisé* nécessite la construction de environs 50 nœuds. La primitive '1' peut faire partie d'une *table* ou une *toilette*. L'exploration en largeur permet à chaque étape de vérifier si les éléments associés forment un symbole ou non. Pour cela à chaque étape l'analyseur obtient un score lié à la reconnaissance graphique du symbole et continue la construction de l'arbre en consommant de nouvelles primitives. Nous obtenons à la fin six hypothèses dont quatre ne correspondent à aucun symbole. Les deux hypothèses

restantes (table et toilette) correspondent à des hypothèses structurellement et graphiquement validées. Il s'agit d'une reconnaissance structurelle et graphique simultanée.

En utilisant *IMISketch optimisé* à chaque nœud créé, l'analyseur vérifie la reconnaissance structurelle et la reconnaissance graphique. Pour cela, nous obtenons des hypothèses qui ne vérifient pas à la fois structurellement et graphiquement un symbole.

Avec *IMISketch hybride*, l'analyseur reconnait un symbole d'une façon séquentielle. Il commence par vérifier le contexte structurel d'un symbole. Une fois que le symbole est vérifié structurellement, l'analyseur le vérifie graphiquement. Pour cela, l'arbre d'analyse est construit en profondeur jusqu'à obtenir un symbole bien vérifié structurellement, puis en largeur, pour d'une part vérifier ce symbole graphiquement ou bien détecter structurellement un nouveau symbole.

Nous allons voir qu'en utilisant *IMISketch hybride*, nous garantissons la présence des deux hypothèses transformant la primitive '1' en une *table* ou en une *toilette*. La figure 4.17 montre l'arbre de construction hybride permettant l'interprétation de la primitive '1' en un mobilier (*toilette* ou *table*). Chaque *séquence mobilier* définie deux zone de recherche : *extrémité réduite* et *extrémité étendue* (figure 4.18). Le concepteur considère que les éléments se trouvant dans la zone '*extrémité réduite*' appartenant au même mobilier car ils sont très proches les uns aux autres. L'ordre d'interprétation des éléments n'a pas d'impact sur le résultat final de l'interprétation. Pour cette raison, la construction de l'arbre est fait en *profondeur*. Les primitives '1', '2', '3', '4' sont interprétées alors en une *séquence mobilier* en adaptant une construction en profondeur. L'utilisation de l'exploration en 'profondeur' dans une règle à «faible risque». Une fois que plus d'élément n'est contenu dans l'*extrémité réduite*, l'analyseur continue la construction de l'arbre d'analyse en appliquant les règles de productions appelées par la zone *extrémité étendue*. Selon les règles décrites dans le tableau 4.3, les règles appelées par la zone *extrémité étendue* permet de soit transformer une *séquence mobilier*, soit étendre la *séquence mobilier* s'il reste des primitives dans la zone *extrémité étendue*.

Notons que, la présence d'un élément dans l'*extrémité réduite* passe également par l'*extrémité étendue*. De plus, chaque *séquence mobilier* dans le document pourrait être transformé en un mobilier. Normalement, trois règles sont applicables transformant cet élément. Deux règles en *'largeur'* correspondant à la règle transformant la *séquence mobilier* en une primitive et la règle déclenchée par l'*extrémité étendue* et un règle en *profondeur* correspondant à la règle déclenchée par l'*extrémité réduite* sont normalement applicables, mais en réalité uniquement la règle en *profondeur* qui sera appliquée

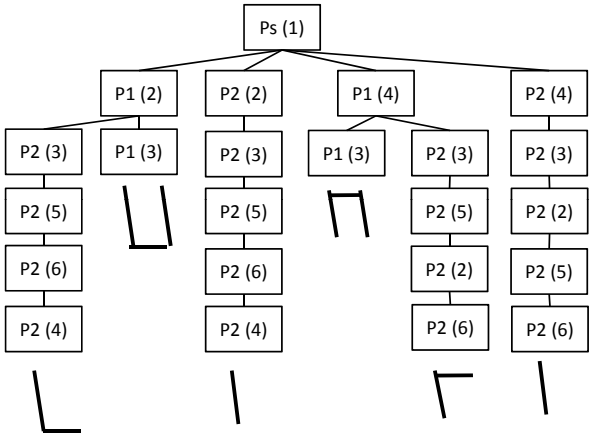


Figure 4.15 – Arbre d'analyse permettant l'interprétation de la primitive '1' en un mur. Cet arbre d'analyse montre l'impact de l'interprétation de la primitive '1' en un mur sur son voisinage (les primitives '2','3', '4', '5' et '6'). La transformation de la primitive '1' en un mur engendre la transformation des primitives voisines en des murs ou des parties d'ouvrant.

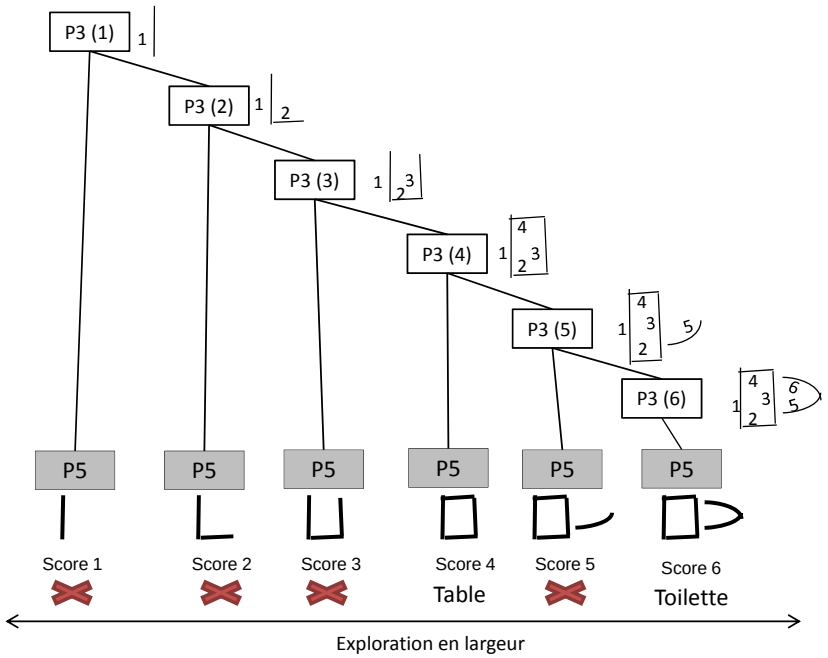


Figure 4.16 – Une partie de l'arbre d'analyse résultant de l'interprétation de la primitive '1' en tant qu'un mobilier (cet arbre d'analyse contient environs 50 nœuds).

car une règle en profondeur est une règle à faible risque et donc le processus d'analyse la met en priorité.

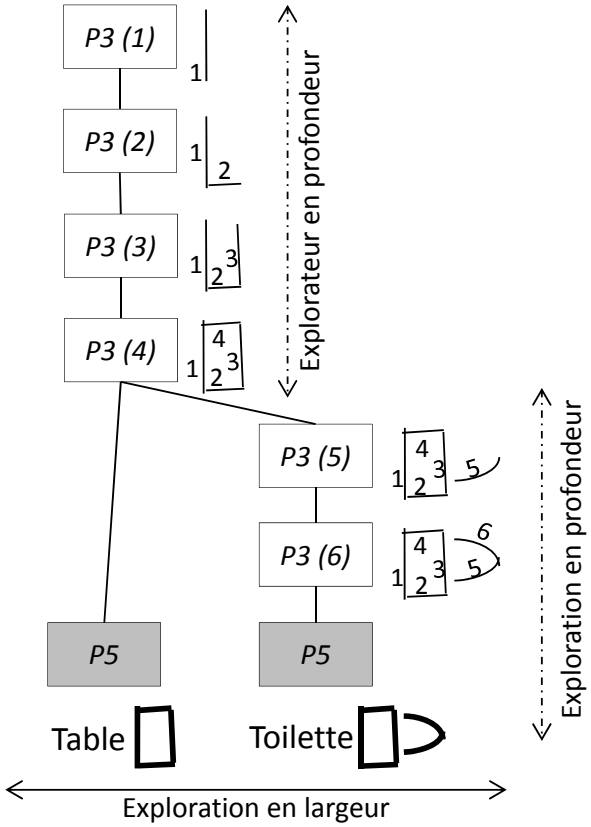


Figure 4.17 – Arbre d'analyse en utilisant l'exploration hybride : transformation de la primitive '1' en une table ou toilette (contient 8 nœuds).

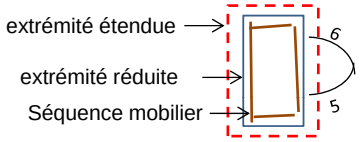


Figure 4.18 – Une *séquence mobilier* ainsi que les deux zones associées : *extrémité réduite* et *extrémité étendue*.

Nous avons présenté dans cette section, des optimisations sur les construction d'analyse afin de garantir à la fois la présence de toutes les hypothèses possibles pour l'interprétation d'un élément et donc la bonne gestion de l'incertitude et réduire la combinatoire engendrée par un tel système. Dans notre méthode IMISketch, il ne s'agit ni d'une méthode d'exploration en largeur ni d'une méthode d'exploration en profondeur mais plutôt d'une méthode hybride respectant des contraintes structurelles présentées dans

la section 4.3.2.2.a. La reconnaissance du symbole avec l'approche hybride est faite en séquentiel, c'est-à-dire que l'analyseur ne passe à la reconnaissance graphique qu'une fois que le symbole a été bien reconnu structurellement. Une fois le symbole reconnu structurellement, il a deux possibilités : soit il ajoute des primitives afin de reconnaitre un autre symbole structurellement, soit il reconnait le symbole graphique. Ces deux hypothèses sont envisageables. Pour cela, il est important de mettre en compétition ces hypothèses.

Cette optimisation peut engendrer une insuffisance d'information sur les hypothèses et donc une éventuelle erreur de reconnaissance. Mais grâce à l'interactivité, cette insuffisance impacte peu sur le résultat final dans la mesure où l'utilisateur peut être sollicité pour valider la bonne hypothèse. L'ensemble de ces optimisations permet d'élaborer une stratégie interactive d'analyse de documents structurés basée sur une exploration hybride de l'arbre d'analyse, en remédiant au problème majeur de l'analyse en largeur qui est l'explosion de la combinatoire.

4.4 Prise de décision

La phase de prise de décision est appelée durant la phase de construction des arbres d'analyse pour attribuer un score à chaque hypothèse et après la phase de construction pour choisir la bonne hypothèse.

4.4.1 Calcul des scores

Chaque hypothèse (branche) possède un score déterminé à partir du formalisme grammatical. Comme décrit dans la section 3.2, les grammaires GMC-PC attribuent un score local à chaque nœud et un score global à toute l'hypothèse. Ce score est déduit à partir des préconditions et des contraintes. Le formalisme grammatical, dans la partie *contraintes*, intègre des classifieurs (décrits dans la section 4.2.2).

4.4.2 Validation de la reconnaissance

La validation de la reconnaissance suit la phase de construction des arbres d'analyse. Le rôle du processus de décision est de valider la bonne hypothèse parmi un ensemble d'hypothèses concurrentes générées par une exploration hybride (section 4.3.2.2.b). Il s'agit donc d'une validation structurelle. Le processus de décision valide aussi la reconnaissance graphique des symboles. On parle d'une validation de la reconnaissance graphique.

4.4.2.1 Validation de la reconnaissance structurelle

Le processus de décision recherche les hypothèses ambigües à partir des hypothèses concurrentes construites dans les arbres d'analyse. Les équations 4.6 et 4.7 décrivent l'algorithme de recherche adopté par le processus de prise de décision.

$$Hypothèses\,ambigües = \{meilleur\,hypothèse\} \cup \{HypothèsesConcurrentesAmbigües\} \tag{4.6}$$

$$\begin{aligned} HypothèsesConcurrentesAmbigües = \{Hypothèse_{i\in n}\}\,/ \\ Score_{meilleurehypothèse} - Score_{hypothèsei} \leq seuil\,d'ambiguïté \end{aligned} \tag{4.7}$$

avec n désigne le nombre d'hypothèses concurrentes.

Deux cas se présentent :

- Si $|Hypothèses\,ambigües| = 1$: *une validation implicite*. L'analyseur est assez confiant pour choisir la bonne racine sans demander à l'utilisateur. Il valide implicitement la racine qui a la branche ayant le score le plus élevé.
- Si $|Hypothèses\,ambigües| > 1$: *une validation explicite*. Le processus de décision n'est pas sûr de prendre la bonne de décision, il s'agit d'un cas d'ambigüité. Il sollicite donc l'utilisateur pour prendre la bonne décision. L'analyseur présente via une interface graphique l'ensemble des hypothèses ambigües et l'utilisateur choisit la bonne parmi la liste proposée.

La décision ne se limite pas seulement à valider la bonne racine, mais peut aussi permettre de valider une partie de la branche (hypothèse), afin d'accélérer l'analyse. En général, si le fils direct d'un nœud est unique, la validation de ce nœud implique automatiquement la validation de son fils direct (algorithme 1).

```
Fonction Prise de décision( Meilleure hypothèse : liste de nœuds) : liste des nœuds
    nœuds_validés : liste de nœuds ;
    nœuds_validés.add(racine de la meilleure hypothèse) ;
    successeur ← nœuds_validés.dernierÉlément.successeur ;
    Tant que (Nombre de successeur == 1) faire
        nœuds_validés.add(nœuds_validés.dernierÉlément) ;
        successeur ← nœuds_validés.dernierÉlément.successeur
    Fait
    Retourner nœuds_validés ;
Fin
```

Algorithme 1: Algorithme de décision d'un ensemble de nœuds

4.4.2.2 Validation de la reconnaissance graphique

Une fois qu'un symbole est défini structurellement, il sera étiqueté par un classifieur. Parfois, le classifieur hésite entre deux ou plusieurs étiquettes pour le même symbole. On utilise alors les mêmes principes que ceux décrits dans la section 4.4.2.1. La seule différence est qu'il ne s'agit plus d'hypothèses structurelles mais plutôt d'hypothèses d'étiquetage. Dans ce cas, l'analyseur lève une ambiguïté de forme et sollicite l'utilisateur pour choisir la bonne étiquette dans la liste des étiquettes ambigües.

A terme, cette sollicitation permet de faire évoluer le classifieur en assurant un apprentissage incrémental. Il permet également d'introduire de nouvelle classe de symbole et donc d'assurer une auto-évolutivité de la méthode.

4.5 Bilan

Dans ce chapitre, nous avons présenté les blocs de notre méthode générique, interactive *IMISketch* pour la rétroconversion des documents structurés. La généricité du système est associée au formalisme grammatical GMC-PC conçu initialement pour l'interprétation en-ligne à la volée. Nous avons adopté ce formalisme pour la rétroconversion hors-ligne des croquis.

L'interactivité du système exige la construction de plusieurs hypothèses d'interprétation concurrentes. En présence de plusieurs primitives dans le document, l'exploration des hypothèses concurrentes engendre une très grande combinatoire que nous avons maîtrisé par un ensemble de techniques dans les différents blocs de notre méthode.

- Choix de primitives : nous avons choisi deux types de primitives : les segments et les polygones. Ce choix permet une représentation grossière pour la reconnaissance structurelle, et une représentation fine utilisée uniquement pour la reconnaissance de forme des symboles.
- Utilisation du contexte local de recherche : nous considérons que l'interprétation d'une primitive dépend uniquement de son voisinage et pas de tout le document. La construction de l'arbre d'analyse concerne uniquement les éléments existants dans le contexte local de recherche.
- Possibilité de valider plusieurs interprétation à la fois : IMISketch peut valider plusieurs règles de production d'une façon simultanée.
- Amélioration du formalisme grammatical GMC-PC. L'exploration *hybride* au sein d'un arbre d'analyse permet de gérer l'incertitude pour metttre en compétition des hypothèses concurrentes et d'éviter la construction des nœuds portant peu d'information sur la prise de décision.

Dans le chapitre suivant de ce manuscrit, nous appliquons notre approche interactive IMISketch sur les plans d'architecture dessinés à main levée.

Quatrième partie

Prototypage et expérimentations

CHAPITRE 5

Prototypage : implémentation d'IMISketch sur les plans d'architecture manuscrits

Dans ce chapitre, nous nous intéressons à la mise en œuvre de l'ensemble de la méthode interactive IMISketch pour la rétroconversion des plans d'architecture manuscrits. Notre méthode interactive est constituée d'un analyseur générique nourri par des connaissances a priori présentant les spécificités des plans d'architecture. Nous nous focalisons sur le dimensionnement de la zone de recherche, sur l'impact de la double représentation (grossière et fine) des primitives pour d'une part, réduire la combinatoire engendrée et d'autre part, garantir la meilleure reconnaissance possible. Nous présentons également l'*interface homme-document* mise en œuvre dans les cas d'ambiguïté.

5.1 Description des règles grammaticales

Les règles grammaticales permettent de décrire le conteneur physique et logique des documents. L'utilisation des grammaires GMC-PC permet de décrire à la fois, une vision globale d'un symbole en s'intéressant à sa position par rapport à son voisinage et une vision locale modélisée par des contraintes.

L'application développée a pour but d'interpréter des plans d'architecture contenant des murs, des ouvrants et des mobiliers. Des exemples de plans d'architectures sont illustrés dans la figure 6.2. L'interprétation des constituants d'un plan prend en compte des spécificités de description telles que :

– les mobiliers dans un plan peuvent être attachés aux murs ;

– le mobilier est à l'intérieur du plan d'architecture ;
– un mobilier est un ensemble de primitives (généralement beaucoup plus deux primitives dans notre cas) interconnectées ou très proches ;
– un ouvrant est un ensemble de primitives (généralement plus de deux) interconnectées ou très proches, qui repose sur un support, généralement, deux murs colinéaires d'un coté et de l'autre de l'ouvrant ;

Notre objectif est alors de décrire des règles de grammaire permettant d'interpréter les primitives (segments et polygones) extraites des plans de la façon suivante :

– un mur est une primitive ;
– un ouvrant est un ensemble de primitives ;
– un mobilier est un ensemble de primitives.

Afin de remédier aux difficultés citées ci-dessus, nous considérons que, à l'extrémité d'un mur, nous pouvons trouver un autre mur, un ouvrant ou un mobilier. Nous considérons également qu'à l'initialisation du processus la primitive la plus longue fait partie d'un mur ou d'un mobilier. Le tableau 5.1 illustre l'ensemble des règles grammaticales utilisées pour ce prototype.

Règle	Éléments créés	Éléments consommés	Type d'exploration
RetroMurDepart	Mur départ	Primitive le plus longue dans un document	Largeur
RetroMobilierDepart	SequenceMobilier	Primitive le plus longue dans un document	Largeur
RetroSequenceMobilierDepart	SequenceMobilier	Primitive à l'extrémité d'un mur	Largeur
RetroSequenceMobilier	SequenceMobilier	Primitive à l'extrémité **réduite** d'une séquence mobilier	Profondeur
RetroSequenceMobilierExtrémitéEtendue	SequenceMobilier	Primitive à l'extrémité **étendue** d'une séquence mobilier	Largeur
RetroMobilier	Mobilier	SequenceMobilier qui correspond graphiquement à un mobilier	Largeur
RetroMur	Mur	Primitive à l'extrémité d'un mur	Largeur
RetroDepartSequence	SequenceOuvrant	Primitive à l'extrémité d'un mur	Largeur
RetroSequenceOuvrant	SequenceOuvrant	Primitive à l'extrémité d'une sequenceOuvrant	Largeur
RetroOuvrantPrimitive	Deux murs et un ouvrant	SequenceOuvrant entre un mur et une primitive colinéaires et semblable à un ouvrant	Largeur
RetroOuvrantMur	Deux murs et un ouvrant	SequenceOuvrant entre deux murs colinéaires et semblable à un ouvrant	Largeur

Tableau 5.1 – Synthèse des règles grammaticales utilisées

Nous commençons l'analyse par les règles décrites dans les figures 5.1 et 5.2. Le concepteur considère que la primitive la plus longue dans le document peut être soit

un mur soit le début d'un mobilier. Par conséquent, ces règles (figures 5.1 et 5.2) sont étiquetées *'largeur'* afin de mettre en action une analyse en largeur autorisant la mise en concurrence de ces deux hypothèses.

Le mur créé ouvre des zones à ses extrémités pour piloter l'analyse en cherchant la présence d'un autre mur ou d'un début d'ouvrant (sequenceOuvrant) ou d'un début de mobilier (sequenceMobilier). Pour cela, ses postconditions font appel aux règles décrites dans la figure 5.7 pour la recherche d'un mur à partir d'une primitive (segment ou polygone), la figure 5.8 pour la recherche d'une partie d'un ouvrant et la figure 5.3 pour le début d'un mobilier. Ces hypothèses sont toutes possibles à l'extrémité d'un mur, pour cela le concepteur choisit d'étiqueter l'exploration en *'largeur'*.

Nom règle : RetroMurDépart

Mur : mRes $\rightarrow$ **(Primitive : pr1** avec { **Type d'exploration :**$'largeur'$

Préconditions :

$\{Document[dans]pr1[tous]\}$

Contraintes :

plusLongSegment($pr1$)

Postconditions :

$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [Mur \rightarrow pr1]\}^* \&$
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [Mur \rightarrow pr1]\}^* \&$
$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow pr1]\}^* \&$
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow pr1]\}^* \&$
$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [SequenceOuvrant \rightarrow pr1]\}^* \&$
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [SequenceOuvrant \rightarrow pr1]\}^*$

Figure 5.1 – Production GMC-PC pour la composition d'un mur. Le mur créé déclenche des zones de recherche à ses extrémités (*'extrémitéInitialeMur'* et *'extrémitéFinaleMur'*) permettant de piloter la suite de l'analyse.

La début du mobilier crée une zone afin de trouver sa suite et fait appel alors à aux règles décrites dans les figures 5.4 et 5.5. Vu qu'un mobilier est un ensemble de primitives interconnectées ou très proches, le concepteur ignore l'ordre d'interprétation de cet ensemble. Il considère que toutes les primitives très proches et interconnectées vont appartenir au même mobilier et donc elles peuvent être interprétées dans n'importe quel ordre. Pour cela, il donne l'étiquette *'profondeur'* à la règle décrite dans la figure 5.4, afin d'enchainer une construction en profondeur de l'arbre d'analyse. L'analyseur continue à consommer les primitives proches en appliquant la règle décrite dans la figure 5.4. Cette règle n'est pas mise en compétition avec les règles décrites dans les figures 5.5 et 5.6 car elle est étiquetée en *profondeur*. Autrement dit, le passage d'une

Nom règle : RetroMobilierDépart
SequenceMobilier : seqRes → **Segment : s** avec { **Type d'exploration :**$'largeur'$
Préconditions :
$\{Document[dans]t[tous]\}$
Contraintes :
plusLongSegment(s) &
inclusDansLePlan(s)
Postconditions :
$\{seqRes[extrémité\,réduite](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow seqRes, pr1]\}^*$ &
$\{seqRes[extrémité\,étendue](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow seqRes, pr1]\}^*$

Figure 5.2 – Production GMC-PC pour la composition d'une partie d'un mobilier atttachée à un constituant du plan. La sequenceMobilier crée la zone (*'extrémité'*) pour rechercher la suite et la fin du mobilier.

séquence mobilier à un mobilier ne pourra se faire que si cette règle (en profondeur) n'est plus applicable.

Nom règle : RetroSequenceMobilierDepart
SequenceMobilier : seqMobRes → **Primitive : pr** avec { **Type d'exploration :**$'largeur'$
Préconditions :
$\{Mur : m[extrémitéInitialeMur]pr[premier]\}$ *ou*
$\{Mur : m[extrémitéInitialeMur]pr[dernier]\}$ *ou*
$\{Mur : m[extrémitéFinaleMur]pr[premier]\}$ *ou*
$\{Mur : m[extrémitéFinaleMur]pr[dernier]\}$
Contraintes :
InclusDansLePlans(pr)
Postconditions :
$\{seqRes[extrémité](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow seqRes, pr1]\}^*$

Figure 5.3 – Production GMC-PC pour le début d'un mobilier : cette règle déclenche la recherche d'un mobilier à l'extrémité d'un mur. Elle vérifie si la primitive est incluse dans le plan et déclenche la règle décrite dans la figure 5.4 pour chercher la suite du mobilier. Cette règle est appelée si une primitive est à l'extrémité d'un mur.

La recherche d'un ouvrant à l'extrémité d'un mur passe par l'application de la règle décrite dans la figure 5.8. Cette règle déclenche la règle présentée dans la figure 5.9. Cette règle est appliquée en 'largeur' permettant de mettre en compétition les hypothèses d'ouvrant concurrentes. Cette règle déclenche deux règles pour la suite d'analyse.

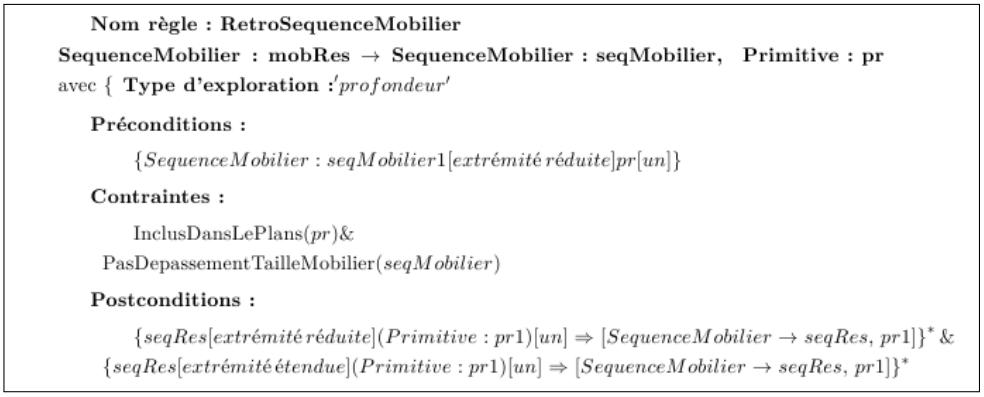


Figure 5.4 – Production GMC-PC pour la suite d'un mobilier : cette règle déclenche la recherche d'une suite de mobilier et permet de chercher la suite du mobilier grâce aux postconditions qui implémentent une récursivité

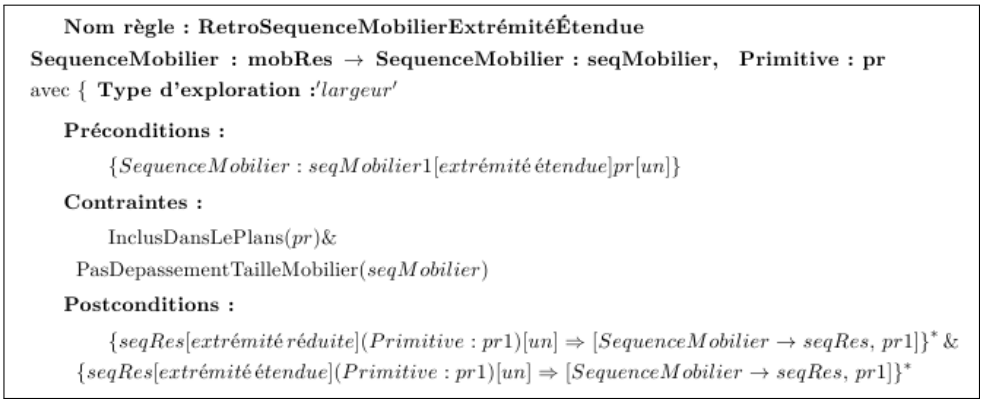


Figure 5.5 – Production GMC-PC pour la suite d'un mobilier : cette règle est appelée si la primitive 'pr' passe par la zone *extrémité étendue* et la règle 5.4 n'est pas applicable.

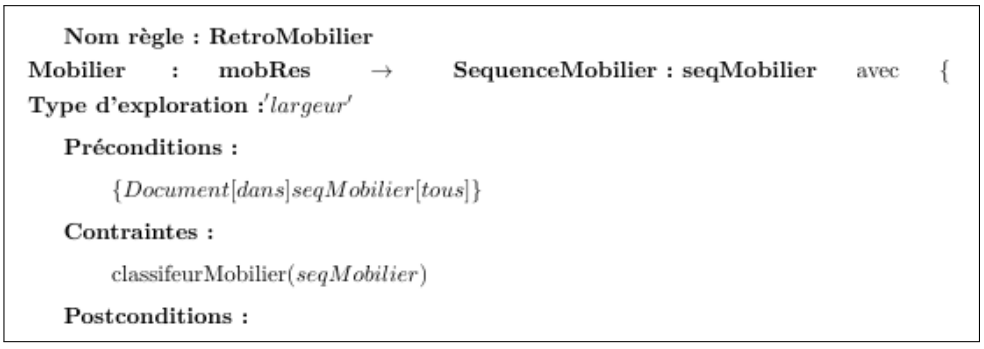


Figure 5.6 – Production GMC-PC pour la composition d'un mobilier. Cette règle est appliquée si l'analyseur ne peut pas appliquer la règle décrite dans la figure 5.4 étiquetée *'profondeur'* et aucune règle est appliquée en *profondeur*

Ces règles permettent de suivre la *'sequenceOuvrant'* (figure 5.9) ou de transformer la *'sequenceOuvrant'* en un ouvrant (figures 5.10 et 5.11)

Nom règle : RetroMur
Mur : mRes → **Primitive : pr** avec { **Type d'exploration :***'largeur'*

Préconditions :

$\{(Mobilier : mob1)[zonePérimètreMobilier]pr[un]\}$ *ou*
$\{(Mur : m1)[extrémitéInitialeMur]pr[premier]\}$ *ou*
$\{(Mur : m1)[extrémitéFinaleMur]pr[dernier]\}$ *ou*
$\{(Ouvrant : seq1)[zonePérimètre]pr[un]\}$

Contraintes :

murSuffisammentLong(pr)

Postconditions :

$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [Mur \rightarrow pr1]\}^*$ &
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [Mur \rightarrow pr1]\}^*$ &
$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow pr1]\}^*$ &
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [SequenceMobilier \rightarrow pr1]\}^*$ &
$\{mRes[extrémitéInitialeMur](Primitive : pr1)[un] \Rightarrow [SequenceOuvrant \rightarrow pr1]\}^*$ &
$\{mRes[extrémitéFinaleMur](Primitive : pr1)[un] \Rightarrow [SequenceOuvrant \rightarrow pr1]\}^*$ &

Figure 5.7 – Production GMC-PC pour la composition d'un mur

Nom règle : RetroDepartSequence
SequenceOuvrant : seqRes → **Primitive : pr** avec { **Type d'exploration :***'largeur'*

Préconditions :

$\{(Mur : m1)[extrémitéInitialeMur]pr[premier]\}$ *ou*
$\{(Mur : m1)[extrémitéFinaleMur]pr[dernier]\}$ *ou*

Contraintes :

PrimitiveNonParallèles(pr)

Postconditions :

$\{seqRes[extrémitéSequenceOuvrant](Primitive : pr1)[un] \Rightarrow [SequenceOuvrant \rightarrow seqRes, pr1]\}^*$

Figure 5.8 – Production GMC-PC pour le début d'un ouvrant : cette règle déclenche la recherche d'un ouvrant à l'extrémité d'un mur. Elle vérifie si la primitive est incluse dans le plan et déclenche la règle décrite dans la figure 5.4 pour chercher la suite d'ouvrant.

L'analyseur interprète les primitives du document en appliquant les règles modélisées par le concepteur. L'analyse se termine lorsqu'aucune règle de production n'est applicable.

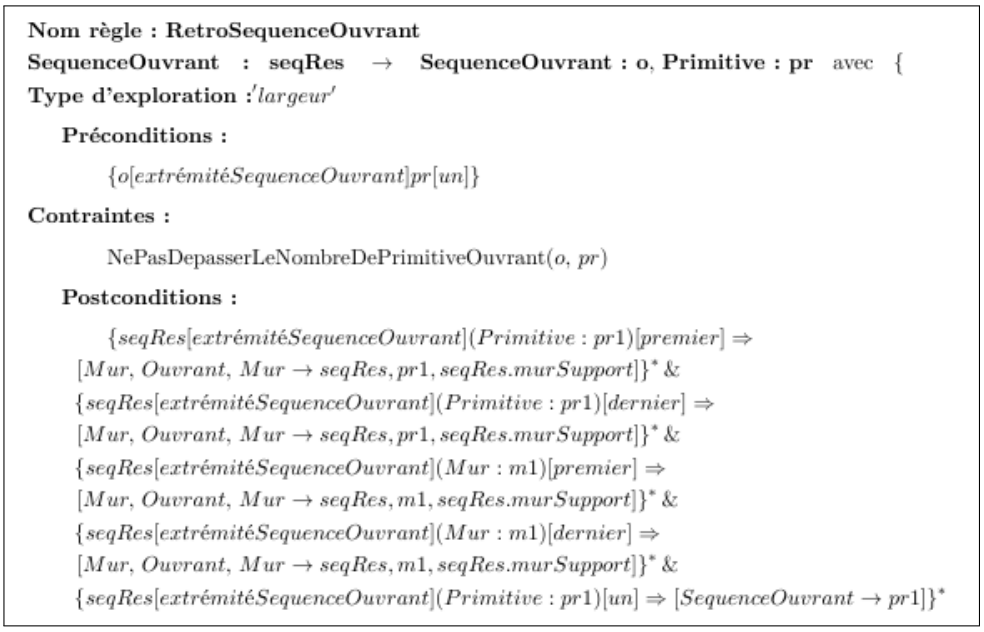


Figure 5.9 – Production GMC-PC pour la suite d'ouvrant : cette règle déclenche la recherche d'une suite d'ouvrant et permet de chercher la suite d'ouvrant grâce aux postconditions qui implémentent une récursivité

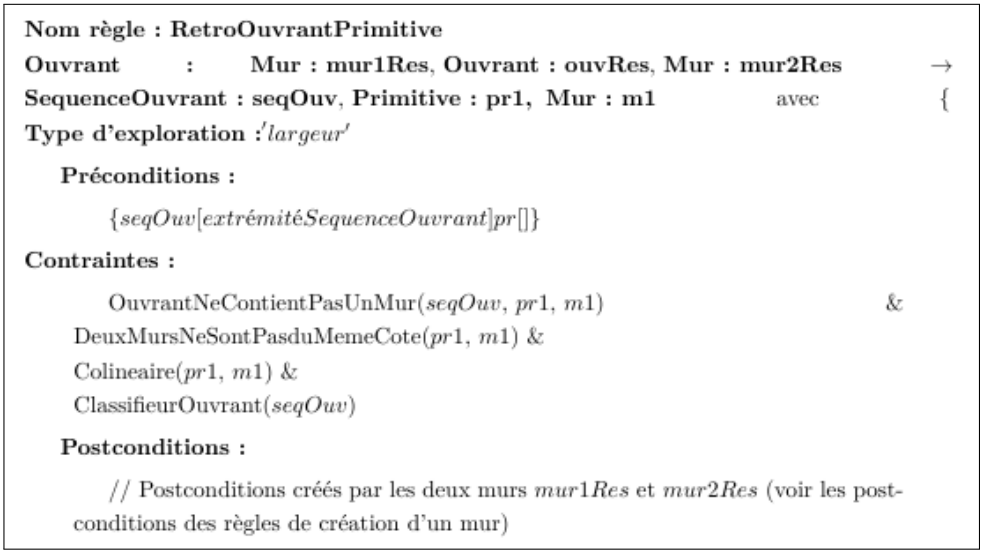


Figure 5.10 – Production GMC-PC pour la transformation d'une SequenceOuvrant en un ouvrant. Cette règle fait appel au classifieur 'ClassifieurOuvrant' qui va permettre de valider l'hypothèse structurelle d'ouvrant.

Nom règle : RetroOuvrantPrimitive
Ouvrant : Mur : mur1Res, **Ouvrant : ouvRes**, **Mur : mur2Res** →
SequenceOuvrant : seqOuv, **Mur : m1**, **Mur : m2** avec {
Type d'exploration :$'largeur'$

Préconditions :

{$seqOuv[extrémitéSequenceOuvrant]m1[]$}

Contraintes :

OuvrantNeContientPasUnMur($seqOuv$, $m1$, $m2$) &
DeuxMursNeSontPasduMemeCote($m1$, $m2$) &
Colineaire($m1$, $m2$) &
ClassifieurOuvrant($seqOuv$)

Postconditions :

// Postconditions créés par les deux murs $mur1Res$ et $mur2Res$ (voir les post-conditions des règles de création d'un mur)

Figure 5.11 – Production GMC-PC pour la transformation d'une SequenceOuvrant en un ouvrant

Ces règles permettent l'interprétation de la plupart des croquis de plans d'architecture. Notons cependant qu'elles ne permettent pas de résoudre le cas de mobiliers à l'extérieur du plan d'architecture et les ouvrants qui ne sont pas connectés à deux murs colinéaires.

5.2 Dimension du contexte local de recherche

Ce contexte est une zone d'intérêt spatial du document, centré sur l'élément se trouvant à droite de la règle de production appliquée à la racine, qui a pour but de limiter la combinatoire dûe à l'exploration en largeur de l'arbre d'analyse. Nous avons vu dans le chapitre 4.3.1 que le choix de la dimension de ce contexte local influe sur la combinatoire engendrée, sur la validation de la bonne interprétation et sur la nombre de sollicitations de l'utilisateur. Un contexte local de recherche trop petit risque de diminuer le taux de reconnaissance du document et un contexte local de grande dimension engendre de l'explosion combinatoire. La taille du contexte dépend principalement du domaine applicatif. Nous recommandons une taille du contexte local permettant de reconnaitre le plus grande symbole du document. En effet, la présence de toutes les primitives d'un symbole dans le contexte de local de recherche permettra la reconnaissance du symbole.

Nous nous intéressons dans cette thèse à la rétroconversion de plans d'architecture dessinés à main levée. Nous considérons que la meilleure taille du contexte local pour l'interprétation des plans contenant des murs, des ouvrants (porte, fenêtre, etc.) et des

mobiliers (lavabo, lit, etc.) correspond à la plus grande taille entre la taille maximale d'un ouvrant et celle d'un mobilier. Cette taille permet d'avoir un contexte suffisant pour décrire toutes les hypothèses d'interprétation possibles pour l'élément 'primitive' associé à la racine. Les figures 5.12 et 5.13 présentent la dimension du contexte local au sein d'une partie de plan d'architecture constitué d'un mur et d'un ouvrant. Les primitives extraites sont de trois segments et un polygone. Le contexte local est piloté par les éléments de la partie droite de la règle de production appliqué à la racine de l'arbre d'analyse.

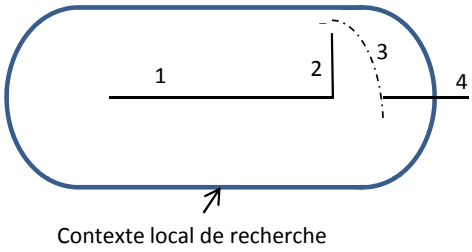


Figure 5.12 – Contexte local de recherche centré par la primitive '1' à l'étape s

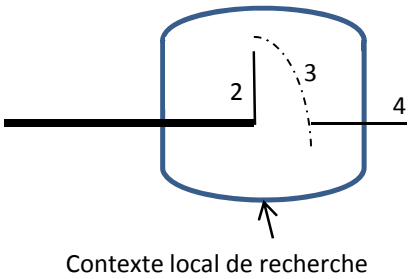


Figure 5.13 – Contexte local de recherche centré par la primitive '2' à l'étape s+1

Si le contexte local de recherche est plus petit que la taille recommandée, tous les symboles de taille supérieure au contexte local seront mal reconnus. En effet, l'arbre d'analyse associé à l'interprétation de ces symboles ne contient pas les hypothèses des symboles et donc ces hypothèses ne seront pas retenues par le processus de décision. La figure 5.14 illustre un exemple de taille de contexte local insuffisante pour la reconnaissance du document et la figure 5.15 présente la conséquence résultante de ce mauvais choix. En effet, au lieu de trouver une porte entre deux murs, le processus d'analyse a reconnu quatre murs.

Si le contexte local est plus grand que la taille recommandée, le contexte local englobera toutes les composantes du document et donc optimisera potentiellement le

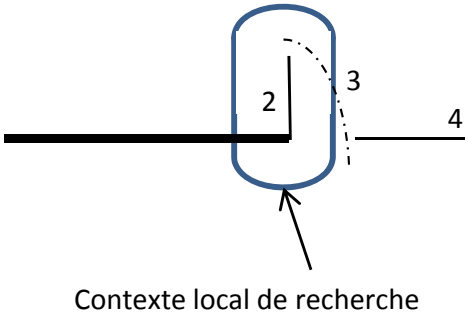


Figure 5.14 – Contexte local de recherche insuffisant pour reconnaitre le symbole 'porte'

problème d'interprétation locale du document. Par contre, la présence d'autant d'éléments dans le contexte local engendrera de la combinatoire.

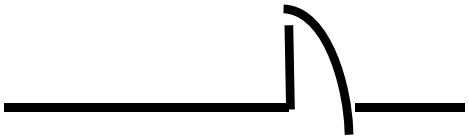


Figure 5.15 – Interprétation de quatre murs au lieu de deux murs et une porte

5.3 Impact de l'utilisation des polygones

Nous avons présenté dans la section 4.1 les deux types de primitives utilisées (polygones et segments) dans notre méthode interactive IMISketch. Pour montrer l'intérêt d'utiliser les polygones comme primitive pour réduire la combinatoire, nous comparons un premier lot de règles de production pour l'interprétation des primitives composées uniquement des segments (tableau 5.2) et un deuxième ensemble de règles avec deux types de primitives : les segments et les polygones. Ce deuxième jeu de règles de production contient les même règles (tableau 5.2) enrichies par trois nouvelles règles qui permettent d'analyser les polygones (tableau 5.3).

La première expérience applique les règles de production décrites dans le tableau 5.2 à un ensemble de segments extraits d'un plan d'architecture (figure 5.16). Nous obtenons l'arbre d'analyse illustré à la figure 5.18. Dans l'étape suivante, le contexte local est déplacé (figure 5.17). Nous obtenons alors l'arbre d'analyse décrit dans la figure 5.19.

Règle	Éléments créés	Éléments consommés
Ps	Mur départ	**Segment** le plus long dans un document
P1	Mur	**Segment** à l'extrémité d'un mur
P2	Séquence ouvrant	**Segment** à l'extrémité d'un mur ou à l'extrémité d'une séquence ouvrant
P3	Ouvrant	Séquence ouvrant entre deux mur colinéaires ou Séquence ouvrant entre un mur et un **Segment** colinéaire
P4	Porte	Un ouvrant
P5	Fenêtre	Un ouvrant

Tableau 5.2 – Règles d'analyse basées sur les segments

Règle	Éléments créés	Éléments consommés
P6	Mur	**Polygone** à l'extrémité d'un mur
P7	Séquence ouvrant	**Polygone** à l'extrémité d'un mur ou à l'extrémité d'une séquence ouvrant
P8	Ouvrant	Séquence ouvrant entre un mur et un **Polygone** colinéaire

Tableau 5.3 – Règles d'analyse complémentaires basées sur des polygones

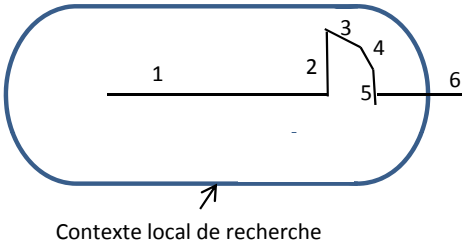


Figure 5.16 – Contexte local de recherche centré par la primitive '1' à l'étape s

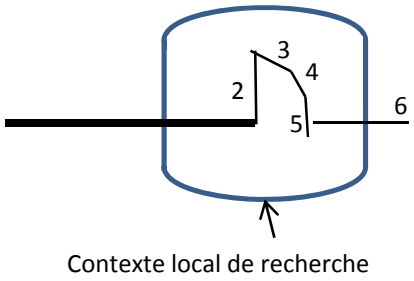


Figure 5.17 – Contexte local de recherche centré par la primitive '2' à l'étape s+1

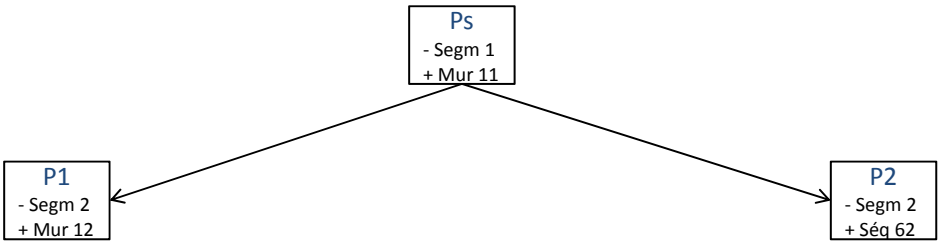


Figure 5.18 – Arbre d'analyse à l'étape s, figure 5.16 avec segments

Bien que cet exemple soit très simple, la validation de l'interprétation des primitives '1' et '2' nécessite la construction de 22 nœuds.

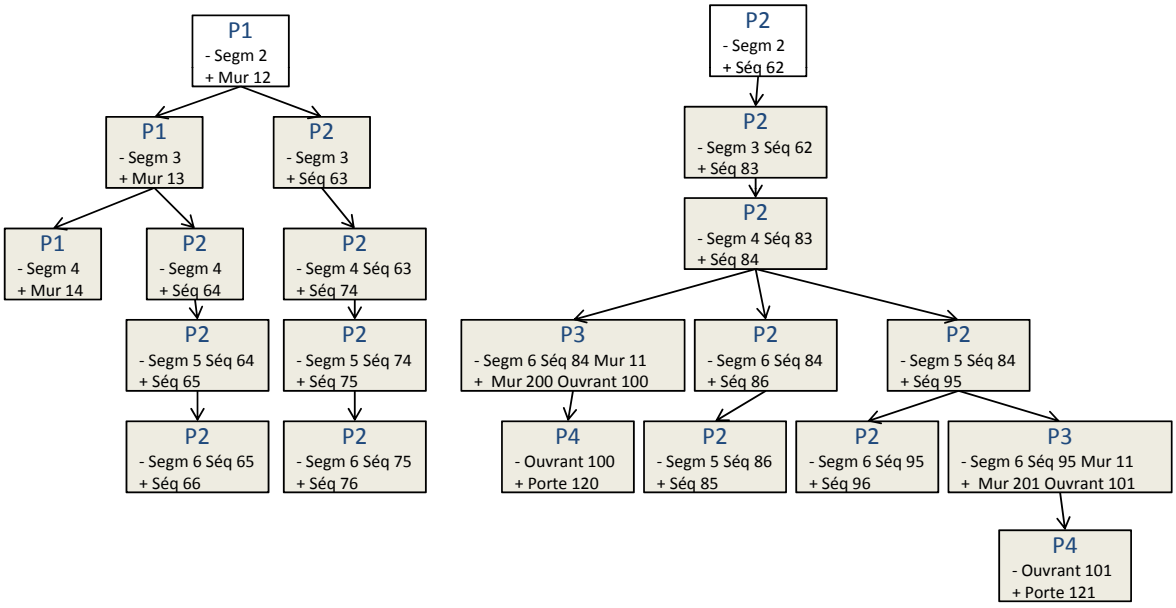


Figure 5.19 – Arbre d'analyse à l'étape s+1, figure 5.17 avec segments. Les nœuds grisés désignent les nouvelles règles de production appliquées en déplaçant le contexte local de recherche.

La deuxième expérience intègre la gestion *des polygones*. La phase d'extraction des primitives produit des segments et des polygones. Les mêmes règles de production utilisées à la première expérience sont appliquées. Nous ajoutons également les règles de production associées à l'interprétation des polygones (tableau 5.3). La figure 5.12 montre les primitives obtenues après la phase d'extraction des primitives : 3 segments (les primitives '1', '2', '4') et un polygone (la primitive '4').

Pour interpréter le segment '1', l'analyseur définit le contexte local (figure 5.12) et construit l'arbre d'analyse associé illustré à la figure 5.20. L'utilisation des polygones comme une primitive, sur cet exemple, permet de passer de 22 nœuds à 12 nœuds ce qui représente un un gain de 46%.

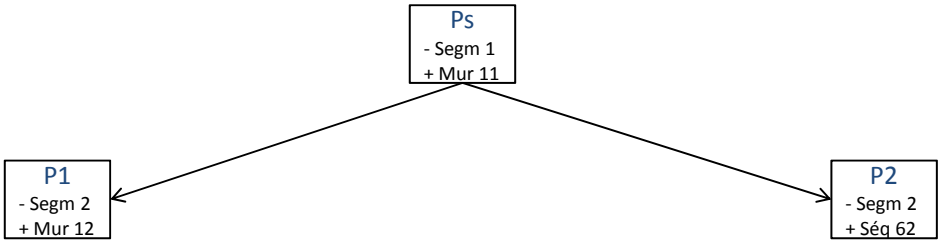


Figure 5.20 – Arbre d'analyse à l'étape s, figure 5.16 avec segments et polygones

Dans l'étape suivante, le segment '2' sera interprété, figure 5.17 avec segments et polygones. La figure 5.12 illustre le positionnement du contexte local associé à ce segment. Les nouveaux arbres d'analyse sont décrits dans la figure 5.21.

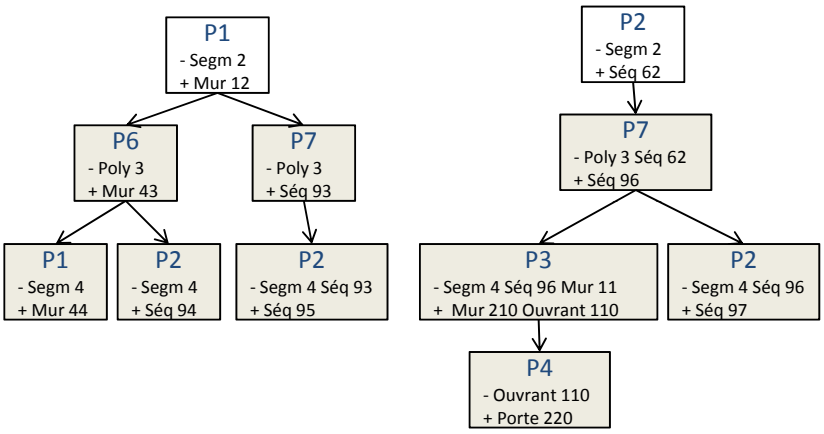


Figure 5.21 – Arbre d'analyse à l'étape s+1. Les nœuds grisés désignent les nouvelles règles de production appliquées en déplaçant le contexte local de recherche.

Nous avons montré dans cette section, l'impact de l'utilisation des polygones sur la construction des arbres d'analyse. Cette intégration de polygone permet des opti-

misations dans le processus de décision. En outre, l'utilisation des polygones minimise la présence d'hypothèses équivalentes et donc allège l'arbre en réduisant le nombre de branches. Ceci permettra de valider une partie entière de la branche pendant la même itération(Section 4.4).

L'utilisation de polygones comme primitive non seulement réduit la combinatoire pour la reconnaissance structurelle, mais garantit également une meilleure reconnaissance de forme pour les symboles obtenus grâce à une représentation plus précise de ses constituants. En effet, une fois le symbole reconnu structurellement, tous les segments formant le symbole (les segments primitives et les segments enchaînés formant le polygone) seront transmis au classificateur afin d'étiqueter le symbole.

Une fois les arbres d'analyse construits, l'analyseur commence la phase finale : la prise de décision. Dans cette phase, l'utilisateur peut être sollicité pour lever les ambiguïtés.

5.4 Les cas d'ambigüité

Dans cette section, nous montrons deux exemples d'ambigüité exigeant une sollicitation de l'utilisateur. Le premier exemple décrit une ambigüité structurelle et le deuxième illustre une ambigüité de reconnaissance de forme.

5.4.1 Ambigüité structurelle

Dans cette section, nous montrons un exemple d'ambigüité structurelle qui exige une sollicitation de l'utilisateur. L'objectif est d'interpréter l'ensemble des primitives extraites à partir d'un plan d'architecture (figure 5.22).

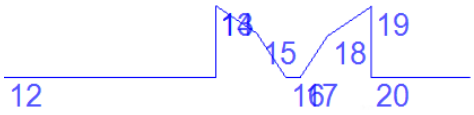


Figure 5.22 – Primitives à interpréter

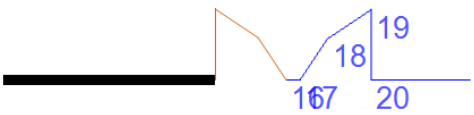


Figure 5.23 – Étape d'interaction

À l'étape illustrée à la figure 5.23, le processus de décision décide de confier la décision à l'utilisateur parce que les deux branches concurrentes sont contradictoires et la différence entre leur deux scores est inférieur au seuil d'ambigüité T_a. En fait, il y a deux façons pour interpréter les primitives de la figure 5.22 : une fenêtre entre deux murs ou 3 murs et deux portes.

L'analyseur commence par interpréter la première primitive (primitive '12') en un mur, ensuite il commence à associer les primitives pour former l'ouvrant. À mi-interprétation le processus de décision se trouve devant deux possibilités : soit transformer la primitive '16' en mur soit la considèrer comme une primitive de la fenêtre. Le processus de décision ne peut pas prendre la décision car il considère que ces deux hypothèses sont applicables, avec un score trop proche.

Il interagit alors en sollicitant l'utilisateur (figure 5.24), et propose de valider la première hypothèse qui est l'hypothèse portant le meilleur score. Cette hypothèse transforme les primitives en 3 murs et 2 portes. Si l'utilisateur valide cette hypothèse, le document sera reconnu comme deux portes (figure 5.25(a)). Si l'utilisateur refuse la première hypothèse, le système valide implicitement la seconde hypothèse (figure 5.25(b)) et le document sera interprété en deux murs et une fenêtre.

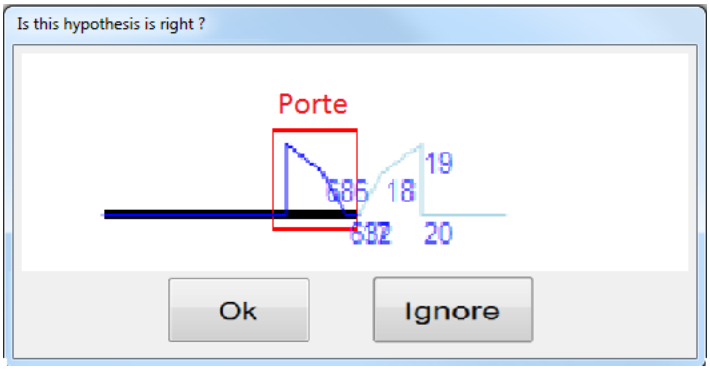
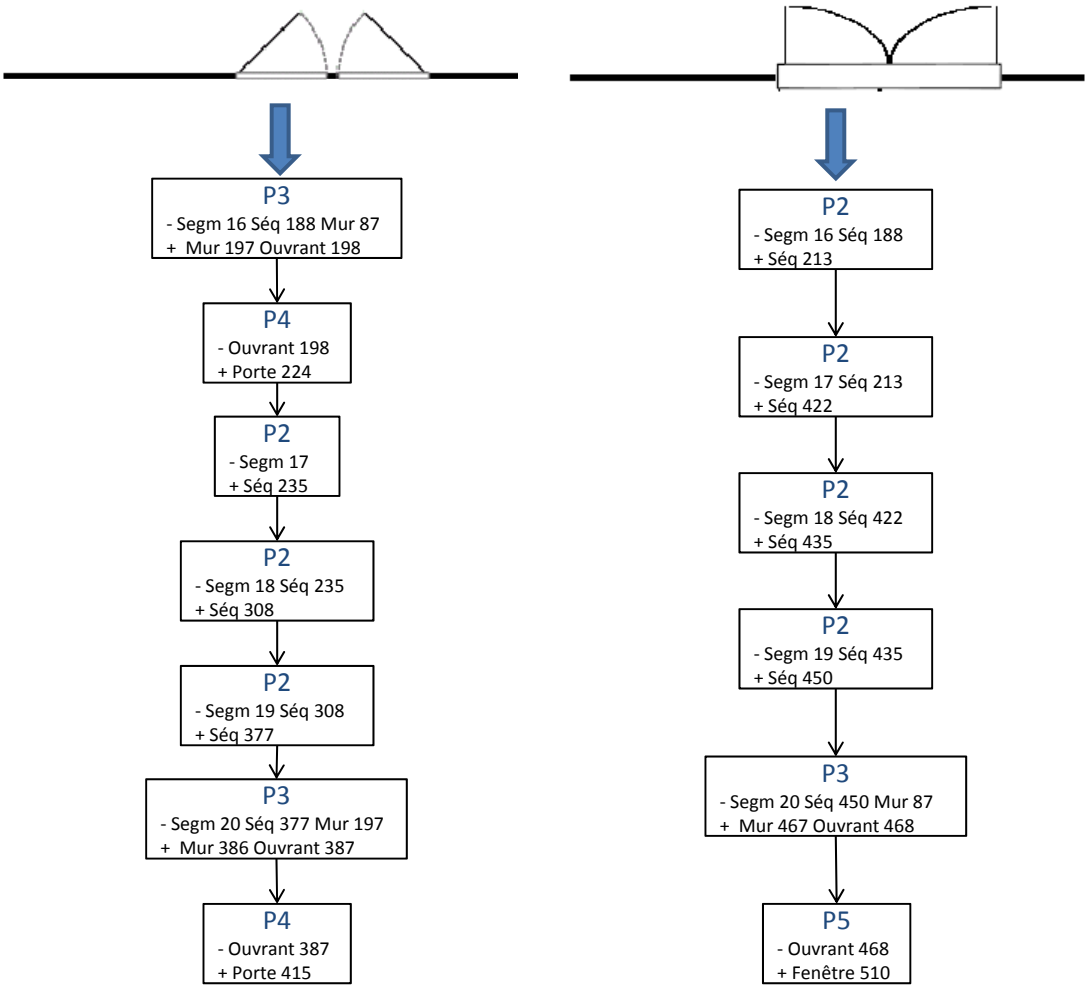


Figure 5.24 – Interaction utilisateur : l'analyseur propose à l'utilisateur la validation de l'hypothèse illustrée dans la figure 5.25(a). Si l'utilisateur ne valide pas cette hypothèse, l'analyseur valide automatiquement l'hypothèse présentée dans la figure 5.25(b)

5.4.2 Ambigüité en reconnaissance de forme

Dans cette section, nous présentons l'interaction en terme de reconnaissance de forme offerte par l'intégration d'un classifieur basé sur un système d'inférence floue



(a) La première hypothèse : transformation des primitives en un mur et deux portes

(b) La deuxième hypothèse : transformation des primitives en un mur et une fenêtre

Figure 5.25 – Exemple d'ambigüité structurelle entre deux hypothèses appliquée à partir des règles décrites dans le tableau 5.2

évolutif [8]. En particulier, nous détaillons quand et comment l'utilisateur peut interagir avec ce classifieur.

Lors de l'analyse, le classifieur est sollicité pour étiqueter un symbole. Le processus de décision utilise le degré de confiance donné par le classifieur pour prendre sa décision. Si le processus de décision considère que le degré de confiance est suffisamment élevé pour prendre la bonne décision, c'est-à-dire que le degré est supérieur à *un rejet de confusion* $> S_a$ il valide l'étiquette proposé par le classifieur. Dans le cas échéant, le processus de décision sollicite l'utilisateur. L'utilisateur est alors en face de quatre possibilités (figure 5.26) :

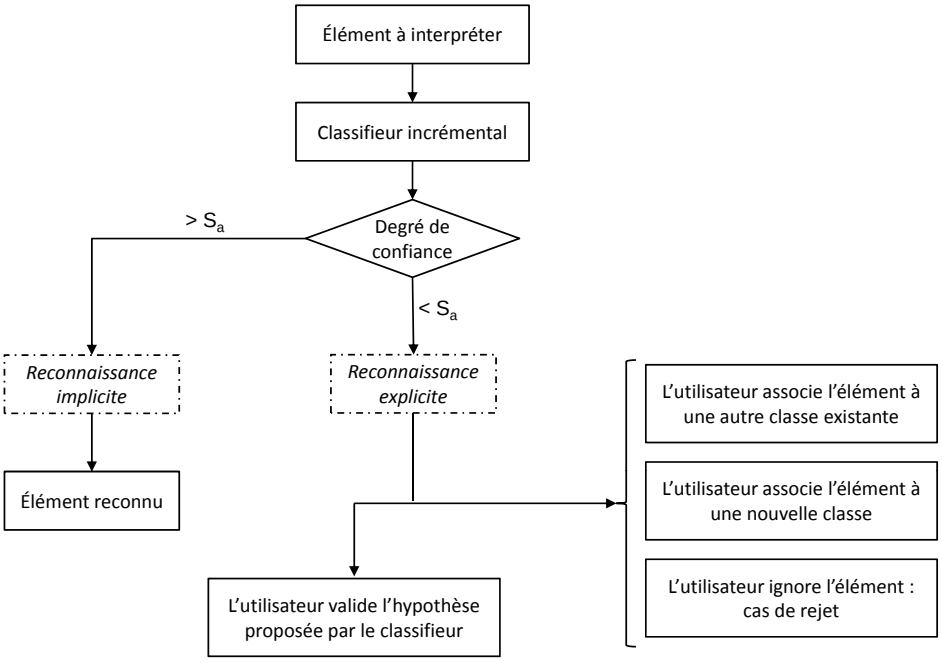


Figure 5.26 – Étape d'interaction

- L'utilisateur valide l'hypothèse proposée par le classifieur malgré le faible degré de confiance donné par le classifieur. Le classifieur va prendre en compte cette réponse pour améliorer le modèle de cette classe.
- L'utilisateur associe le symbole à reconnaître à une autre classe existante dans le classifieur. Le classifieur va apprendre de cette réponse pour réduire la confusion entre deux classes.
- L'utilisateur associe le symbole à une nouvelle classe : l'utilisateur estime que le symbole n'appartient à aucune classe existante. Avec cette nouvelle information, le classifieur va commencer à apprendre une nouvelle classe de symboles.
- L'utilisateur ignore le symbole à reconnaître : c'est le cas de rejet. L'utilisateur estime qu'une fausse interprétation, dûe essentiellement à un problème de segmentation, a eu lieu.

Avec ce processus d'interaction, le classifieur apprend en permanence afin d'améliorer ses interprétations. Plus l'analyse avance, plus le classifieur donne de bonnes classifications et moins l'utilisateur est sollicité. De plus, cet apprentissage incrémental peut faire face à la reconnaissance de nouvelles classes de symboles. C'est un point clé pour absorber la grande variabilité des symboles qui peuvent être présents dans un document.

La figure 5.27 montre un cas où la sollicitation utilisateur est jugée nécessaire pour interpréter un plan d'architecture manuscrit. Dans cette intervention, l'utilisateur se trouve en face de quatre actions possibles décrites précédemment. Le système présente

une interface qui contient l'hypothèse donnée par le classificateur, les autres classes disponibles du classificateur et un champ où l'utilisateur peut ajouter une nouvelle classe. La figure 5.27(a) représente un cas dans lequel l'utilisateur indique que le symbole à reconnaître est une fenêtre classique. La figure 5.27(b) montre un cas dans lequel l'utilisateur associe le symbole d'une nouvelle classe de fenêtres (une fenêtre coulissante). La participation des utilisateurs a un grand impact pour éviter l'accumulation d'erreurs lors de l'étape d'analyse. Cette sollicitation permet au classifieur d'apprendre à partir des confusions des symboles reconnus. L'ajout d'un nouveau symbole permet la création d'une nouvelle classe dans notre système de classification.

Rejet de confusion

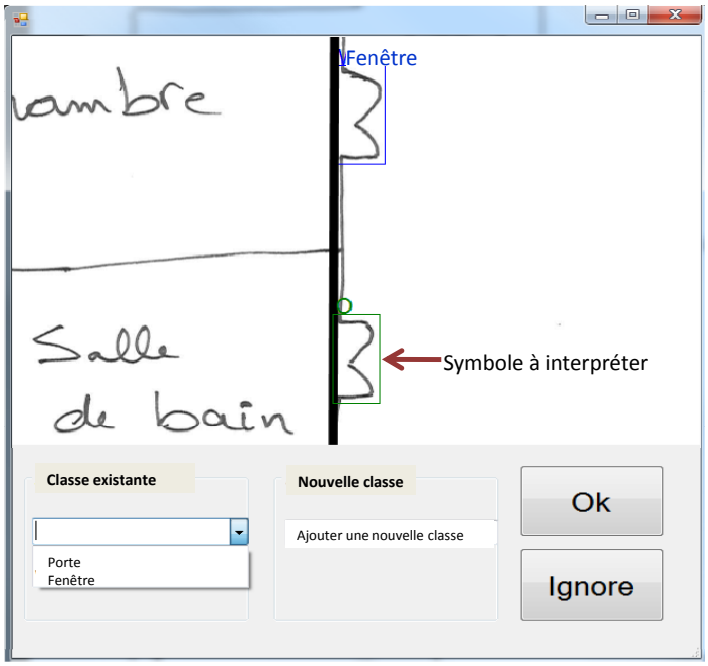
Le but du rejet confusion est d'évaluer la fiabilité du classifieur [8]. Les erreurs sont dues à la présence de deux classes dont les scores sont proches.

Pour formaliser le rejet de confusion, nous utilisons la notion de fonctions de fiabilité. Dans notre cas, la fonction de fiabilité ($\psi(X)$) représente le degré de confusion sur l'étiquetage de l'exemple X :

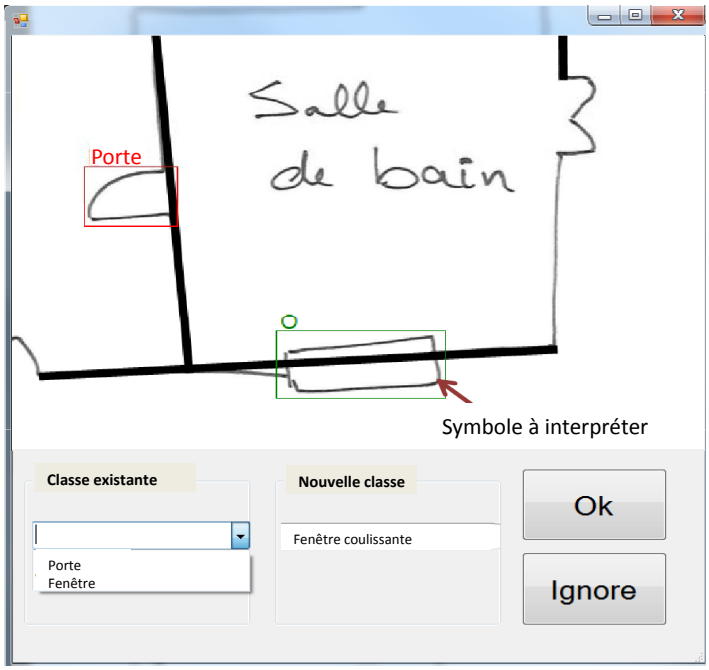
$$\psi(X) = (Sc_1(X) - Sc_2(X)/Sc_1(X) \tag{5.1}$$

où $Sc_1(X)$ est le score obtenu pour la meilleure classe et $Sc_2(X)$ est le score obtenu pour la seconde meilleure classe. Un exemple X est rejeté lorsque le degré de confusion est en dessous d'un seuil spécifique.

Nous avons présenté dans ce chapitre des focus sur l'implémentation des principales parties de note méthode interactive IMISketch. Cette méthode a été appliquée aux plans d'architecture. Nous avons illustré des exemples de règles de production utilisées pour la description des plans d'architecture dessinés à main levée. Nous avons présenté également des interfaces graphiques portant sur les ambigüités structurelles et en reconnaissance de forme. Dans le chapitre suivant, nous évaluons notre méthode interactive IMISketch.



(a) L'utilisateur valide l'interprétation permettant la transformation du symbole en une fenêtre



(b) L'utilisateur valide l'interprétation permettant la transformation du symbole en une nouvelle classe : fenêtre coulissante

Figure 5.27 – Sollicitation utilisateur. Quatre possibilités peuvent exister. L'utilisateur associe l'ensemble de primitives localisé dans la boite 'o' à la bonne classe

CHAPITRE 6

Expérimentations

Dans ce chapitre, nous évaluons les performances des systèmes conçus à l'aide de la méthode interactive IMISketch. Les premières expérimentations sont des tests unitaires sur les contributions de cette thèse. Ces expériences portent essentiellement sur l'apport de l'interactivité au sein d'une méthode de reconnaissance a posteriori et sur la façon de présenter une information à l'utilisateur afin de le mettre dans les meilleures conditions possibles pour bien interagir avec le système. De plus, nous présentons des tests unitaires pour valider les optimisations décrites dans le chapitre 4 afin de réduire la combinatoire. La deuxième partie des expérimentations est considérée comme une évaluation globale de notre méthode interactive d'interprétation de documents structurés. La dernière partie de ce chapitre décrit les travaux à venir en validant, par un test unitaire, la possibilité d'intégrer un classifieur incrémental capable d'apprendre et d'ajouter de nouvelle classes de symboles durant la phase d'analyse.

6.1 Base de données

Pour mener les expériences de cette thèse, nous avons créé deux bases de plans d'architecture de complexité différente. La première, nommée *'base complexe'* est constituée de 24 plans d'architecture manuscrits, dessinés par six personnes. Chaque plan d'architecture est composé d'un ensemble de murs, d'ouvrants de plusieurs types (portes, fenêtres et fenêtres coulissantes) et de mobiliers (évier, lavabo double, lavabo simple, WC, baignoire, table basse, table avec des chaises, lit simple, lit double, douche, canapé). La figure 6.1 présente quelques exemples des plans d'architecture de la base. Le tableau 6.1 reporte le nombre total de symboles au sein des 24 plans d'architecture.

La deuxième base, nommée *'base simplifiée'* est constituée de 69 plans d'architecture dessinés à main levée, de complexité variable, dessinés par une dizaine de personnes. Chaque plan d'architecture est composé d'un ensemble de murs, portes, fenêtres et

(a) (b)

(c)

Figure 6.1 – Exemple de plans d'architecture de la base complexe

Nombre de plan d'architecture	24
Nombre de murs	961
Nombre d'ouvrants	414
Nombre de mobiliers	523

Tableau 6.1 – Propriétés de la base des plans d'architecture : base complexe

fenêtres coulissantes. Cette base de données contient 2641 murs et 1333 ouvrants (555 portes, 377 fenêtres et 401 fenêtres coulissantes) (tableau 6.2). Quelques exemples de plans d'architecture sont illustrés dans la figure 6.2.

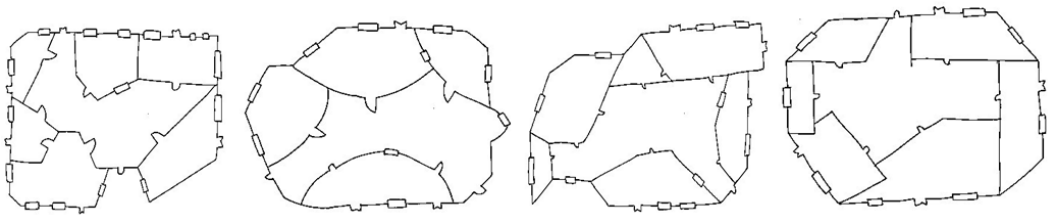


Figure 6.2 – Exemples de plans d'architecture de la base simplifiée

Nombre de plan d'architecture	69
Nombre de murs	2641
Nombre d'ouvrants	1333

Tableau 6.2 – Propriétés de la base des plans d'architecture : base simplifiée

Chaque expérimentation a été appliquée sur l'une de ces deux bases. La *'base simplifiée'* a été utilisée pour montrer l'impact des évolutions de notre méthode interactive IMISketch sur la combinatoire (plus de détail dans la section 6.7). La *'base complexe'* permet de démontrer l'intérêt d'avoir une méthode interactive pour la reconnaissance des plans d'architecture (section 6.2). De plus, cette base est utilisée pour montrer le gain fourni par l'utilisation d'une approche hybride qui garantit la mise en concurrence entre les hypothèses tout en évitant la création d'hypothèses inutiles (section 6.5).

6.2 Test unitaire : apport de l'interactivité pour la reconnaissance de documents structurés

L'intégration de l'utilisateur durant l'analyse permet d'éviter la propagation des erreurs et donc soulage la phase de vérification a posteriori qui pourrait être fastidieuse. Une pré-étude de début de projet faite par l'équipe *LPE* du CRPCC, a consisté à demander à des participants de comparer deux plans disposés côte-à-côte afin de détecter les erreurs. Alors que cette tâche semble simple, seulement 33% des participants sont parvenus à repérer l'ensemble des erreurs. Ceci illustre donc les difficultés d'une correction a posteriori de l'interprétation d'un plan et donc l'intérêt de soulager cette tâche en allant vers une solution interactive de rétro-conversion.

Nous présentons dans cette partie l'impact de la sollicitation de l'utilisateur pour l'interprétation des documents structurés. Nous avons utilisé la *'base complexe'* (avec mobilier). Nous faisons varier le taux d'ambiguïté au delà duquel la sollicitation de l'utilisateur est requise. La figure 6.3 reporte le taux de reconnaissance structurelle avec les différents seuils de sollicitations de l'utilisateur. Cette figure montre l'évolution des taux de reconnaissance par rapport au nombre de sollicitations de l'utilisateur.

Nous remarquons que le meilleur compromis (taux de reconnaissance/sollicitations utilisateurs) est obtenu avec 12 sollicitations d'utilisateur par l'image : cela indique que 4% des interprétations de primitives exigent la sollicitation de l'utilisateur. 49% des sollicitations d'utilisateur sont utiles pour prendre la bonne décision, c'est-à-dire

que l'utilisateur ne valide pas la première hypothèse (la meilleure hypothèse) proposée par l'analyseur. Le taux de reconnaissance structurelle passe de 91% sans sollicitations utilisateur à 96% avec sollicitations utilisateur.

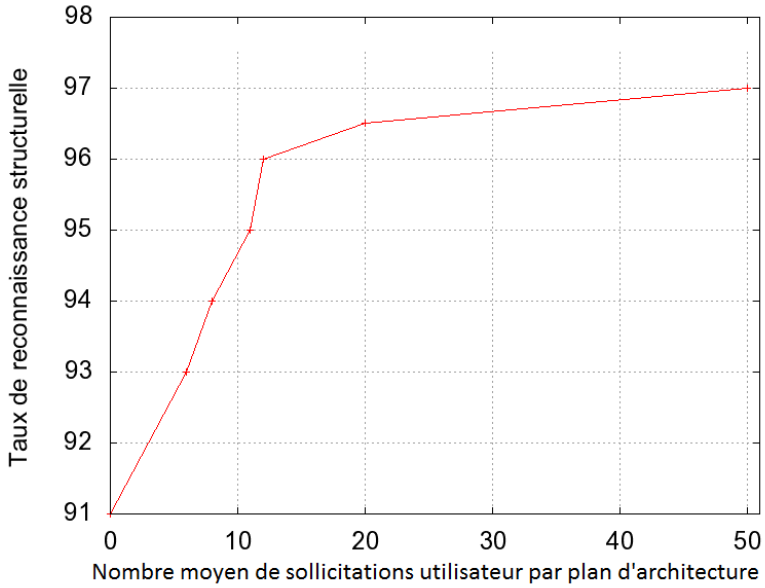


Figure 6.3 – Évolution de la reconnaissance structurelle en variant le seuil d'ambiguïté, et donc le nombre d'interventions de l'utilisateur

Le seuil d'ambiguïté vérifiant le meilleur compromis sollicitation/taux de reconnaissance, déduit de cette expérimentation sera introduit dans les tests unitaires qui suivent et dans le l'évaluation globale du processus d'analyse.

Concernant l'interface homme-machine mise en place, nous avons lancé un travail collaboratif avec l'équipe *LPE* du laboratoire CRPCC, de psychologie cognitive. Les expérimentations portent essentiellement sur la façon de présenter l'information nécessaire à l'utilisateur pour qu'il choisisse la bonne hypothèse.

6.3 Test unitaire : sollicitation de l'utilisateur et interfaçage

Le prototype de notre méthode interactive IMISketch a fait l'objet des expérimentations cognitives au sein de l'équipe *LPE* du laboratoire CRPCC. Ces expérimentations ont porté essentiellement sur la façon de présenter l'information à l'utilisateur pour résoudre une ambiguïté levée par le processus d'analyse. Nous décrivons brièvement

dans cette section ces expérimentations. Tous les détails sont reportés dans la thèse de Sylvain Fleury [1] .

Cette étude a été basée sur deux expérimentations qui ont pour objectif de déterminer l'impact de présentation à l'écran des plans d'architecture sur la qualité de leur correction par des utilisateurs.

6.3.1 Expérimentation : impact de la présentation de l'information sur la qualité de leur correction par des utilisateurs

Pendant cette expérimentation, les participants doivent comparer le plan original avec son format numérique interprété afin de détecter les erreurs réalisées.

Cette première expérimentation a porté sur 54 volontaires (19 hommes et 35 femmes).

Les épreuves de comparaison de plans ont été effectuées avec un prototype de notre méthode IMISketch sur trois plans différents contenant des murs, des portes, des fenêtres et des fenêtres coulissantes. Chaque participant devait ainsi comparer trois paires de plans successivement dans une des trois conditions expérimentales : *séparée*, *intégrée* et *séquentielle*.

6.3.1.1 Condition *séparée*

Le plan manuscrit s'affiche à l'écran et aucune interaction n'est possible tant que la phase d'analyse est en cours. Ensuite, le plan rétroconverti apparait à côté du plan manuscrit. Les participants peuvent alors entourer les erreurs. La figure 6.4 illustre un exemple de plan traité avec la condition séparée.

6.3.1.2 Condition *intégrée*

La condition intégrée est similaire à la condition séparée puisque seul le plan manuscrit apparait à l'écran pendant toute la durée de l'interprétation. En revanche, au lieu de faire apparaitre le format numérique du plan analysé à côté du plan manuscrit, il s'affiche par-dessus du plan manuscrit à la fin du processus de rétroconversion. La figure 6.5 illustre un exemple de plan traité avec la condition intégrée.

1. Doctorant à l'équipe *LPE* qui travaille dans le cadre du projet *ANR MobiSketch*

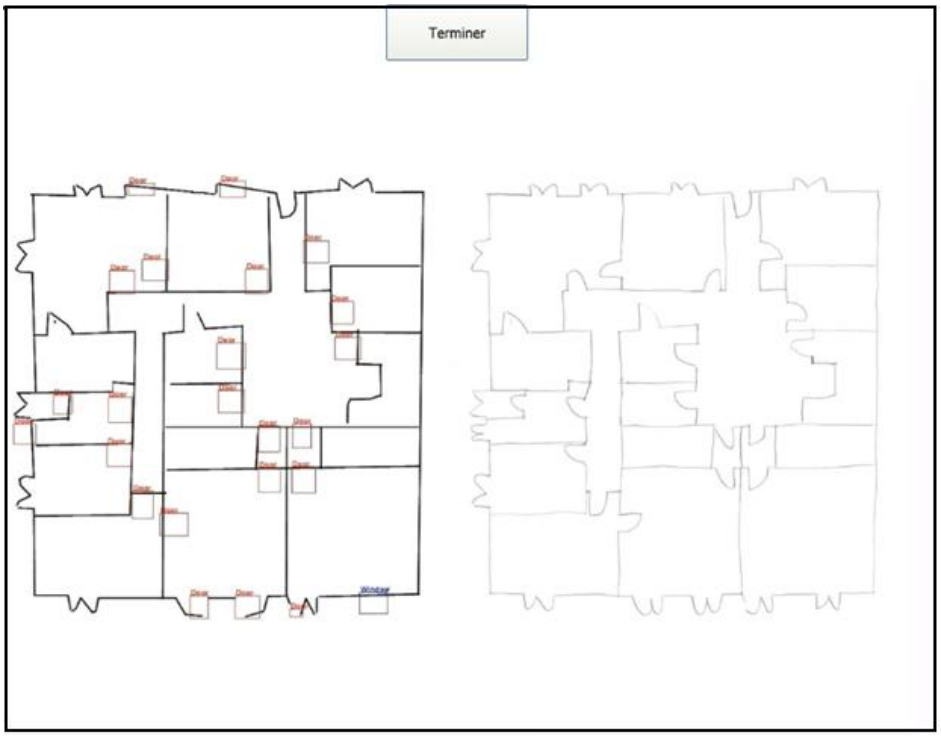


Figure 6.4 – Condition *séparée* : le format interprété est a côté du plan manuscrit

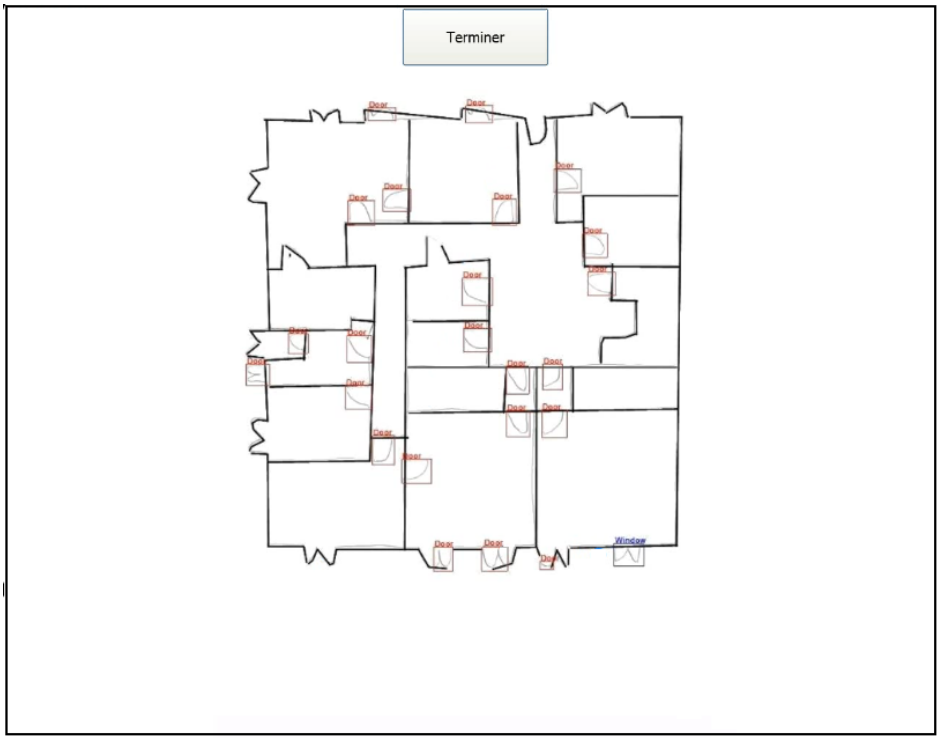


Figure 6.5 – Condition *intégrée* : l'interprétation numérique apparait par-dessus le plan manuscrit à la fin du processus d'analyse

6.3.1.3 Condition *séquentielle*

La condition séquentielle consiste à construire le plan rétroconverti progressivement par-dessus le plan manuscrit et donc le processus d'interprétation est visible en temps réel pour les participants. Un exemple d'interface implémentant cette condition est présenté dans la figure 6.6.

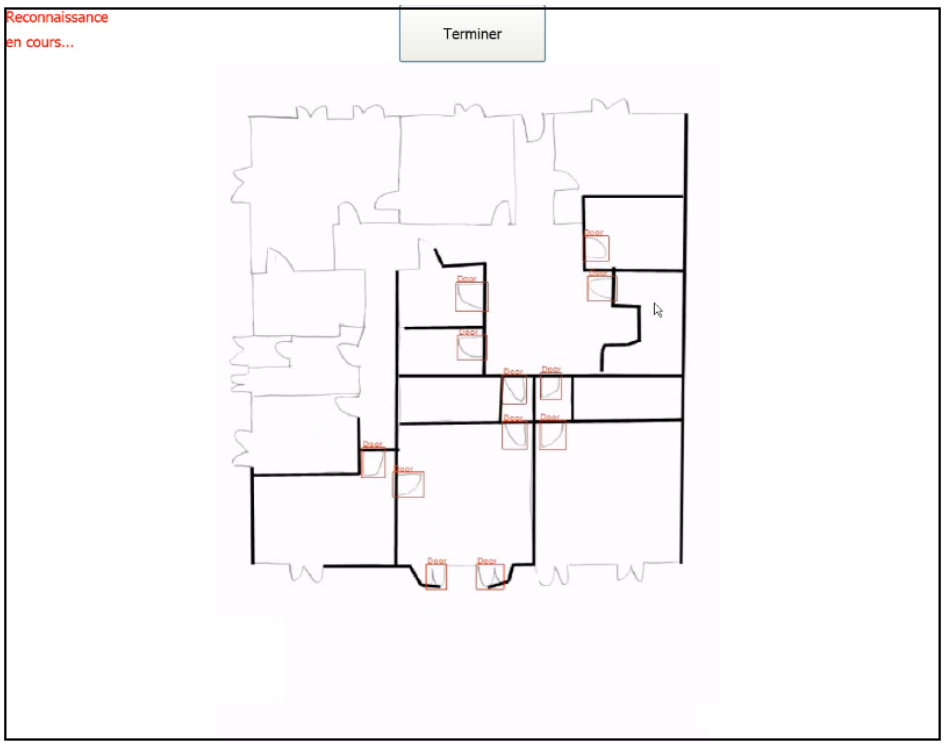


Figure 6.6 – Condition *séquentielle* : le plan interprété apparait par-dessus le plan manuscrit d'une manière progressive

6.3.2 Discussion

Les expériences faites avec ces trois conditions montrent que les participants ont réalisé la tâche dans une durée significativement plus courte avec des plans superposés qu'avec des plans séparés. Dans la condition *séparée*, les participants encodent des informations visuo-spatiales sur un plan, les maintiennent en mémoire jusqu'à avoir trouvé visuellement la zone correspondante sur l'autre plan et effectuent alors la comparaison entre les deux plans (plan manuscrit et format numérique interprété). En revanche, la superposition des plans supprime ces étapes de recherche visuelle et de maintien actif de l'information visuo-spatiale liée essentiellement à l'éloignement. En effet, les participants, dans cette condition, disposent de suffisamment d'informations

en regardant à un endroit donné pour repérer une erreur à cet endroit. À l'opposé, les participants dans le format séparé ont aussi assez d'information, mais ces informations sont dispatchées sur deux plans éloignés.

De plus, cette expérience montre que la condition séquentielle pour l'interprétation du plan améliore l'efficacité du prototype. Cette hypothèse est validée par les résultats puisque le repérage des erreurs est meilleur avec le format séquentiel qu'avec le format *intégré*. Cette expérience a montré que seules 35% des personnes ont réussi à repérer toutes les erreurs en *condition intégrée*. En *condition séparée*, le taux de réussite est légèrement meilleur et atteint 47%. Ce taux est largement plus important avec *la condition séquentielle* dans laquelle on arrive à 78% de taux de réussite complète de repérage de toutes les erreurs.

Ce gain pourrait provenir d'un effet de guidage attentionnel proposant aux participants un ordre d'exploration du plan qui, s'il est suivi, permet de s'assurer que l'ensemble du plan est contrôlé, ce qui n'est pas forcément le cas dans les conditions *séparée* et *intégrée*. Nous choisissons alors d'utiliser la condition séquentielle pour notre méthode interactive *IMISketch*, donc l'analyse apparaitra par-dessus le plan analysé, au fur et à mesure de son interprétation.

6.4 Test unitaire : gestion de la combinatoire liée aux primitives

Dans cette partie, nous présentons l'impact de chaque optimisation sur la réduction de la combinatoire et donc du temps de calcul. Pour cela, nous proposons trois versions de IMISketch (tableau 6.3). La première version, appelèe *IMISketch*, exploite toutes les hypothèses (branches) des arbres d'analyse dans le contexte local de recherche en prenant un seul type de primitive (les segments). La seconde version, appelèe *IMISketch+(Seg)*, intègre toutes les optimisations décrites dans la section 4.3.2.2.a. Cette version interprète l'ensemble de primitives composées uniquement des segments. La dernière version, appelée *IMISketch+(Seg&poly)*, est similaire à la seconde. La seule différence est le type de primitives à interpréter, qui regroupe à la fois les segments et les polygones. Le tableau 6.3 illustre un récapitulatif des trois versions. Les expériences ont été réalisées sur la *'base simplifiée'*. Pour chaque plan d'architecture, nous comparons, pour les trois versions (IMISketch, IMISketch+(Seg), IMISketch+(Seg&poly)), le résultat final de l'interprétation et le temps de calcul obtenu. Le tableau 6.4 montre le taux de reconnaissance des plans d'architecture. La figure 6.7 présente le temps de calcul par image. Nous remarquons que les pics (liés aux explosions combinatoires) ont

Version	Type de primitive	Construction des arbres d'analyse
IMISketch	Segments	Toutes les hypothèses dans le contexte local de recherche
IMISketch + (Seg)	Segments	En utilisant les optimisations de IMISketch optimisé (section 4.3.2.2.a)
IMISketch + (Seg&poly)	Segments et polygones	En utilisant les optimisations de IMISketch optimisé (section 4.3.2.2.a)

Tableau 6.3 – Caractéristiques des versions IMISketch

disparus en utilisant IMISketch+. En ce qui concerne les taux de reconnaissance (le

	IMISketch	IMISketch+ (Seg)	IMISketch+ (Seg&poly)
Taux de reconnaissance	97.96%	98.13%	97.93%
Nombre de symboles mal reconnus	81	74	82
Temps de calcul moyen	11 min 03	4 min 43	2 min 31
Nombre moyen de sollicitations de l'utilisateur	6.07	5.17	3.59

Tableau 6.4 – Taux de reconnaissance des plans d'architecture

tableau 6.4), nous notons que les optimisations pour réduire la combinatoire effectuées par IMISketch n'influent pas sur les performances du système. En effet, nous observons une différence très légère entre IMISketch, IMISketch+(Seg) et IMISketch+(Seg&poly) en terme de taux de reconnaissance. Avec la sollicitation utilisateur, lors de l'analyse, on obtient 97,96% avec IMISketch, 98,13% avec IMISketch+(Seg) et 97,93% avec IMISketch+(Seg&poly) (le tableau 6.4).

Ceci peut être expliqué en considérant les hypothèses proposées par les trois méthodes. Les hypothèses ne sont pas les mêmes. IMISketch produit plus d'hypothèses que IMISketch+(Seg) et IMISketch+(Seg) génère plus d'hypothèses que IMISketch+(Seg&poly). Cela pourrait indiquer qu'il y a plus de chance d'avoir la bonne hypothèse avec IMISketch, mais en même temps les confusions générées sont aussi

potentiellement plus nombreuses. En fin de compte, les résultats en termes de performances de reconnaissance sont très comparables.

En outre, la sollicitation utilisateur passe de 6 interventions moyennes par image dans IMISketch à environ 4 interventions dans IMISketch+(Seg&poly) (tableau 6.4). Cette diminution montre une réduction dans les hypothèses concurrentes qui peuvent conduire à des ambiguïtés. Cela est dû à la présence de la primitive polygone qui réduit le nombre de nœuds par branche et par conséquent, les conflits entre les hypothèses. Les erreurs obtenues (environ 2%) sont dues soit à un mauvais calibrage du contexte local, soit au mauvais dessin de certains symboles.

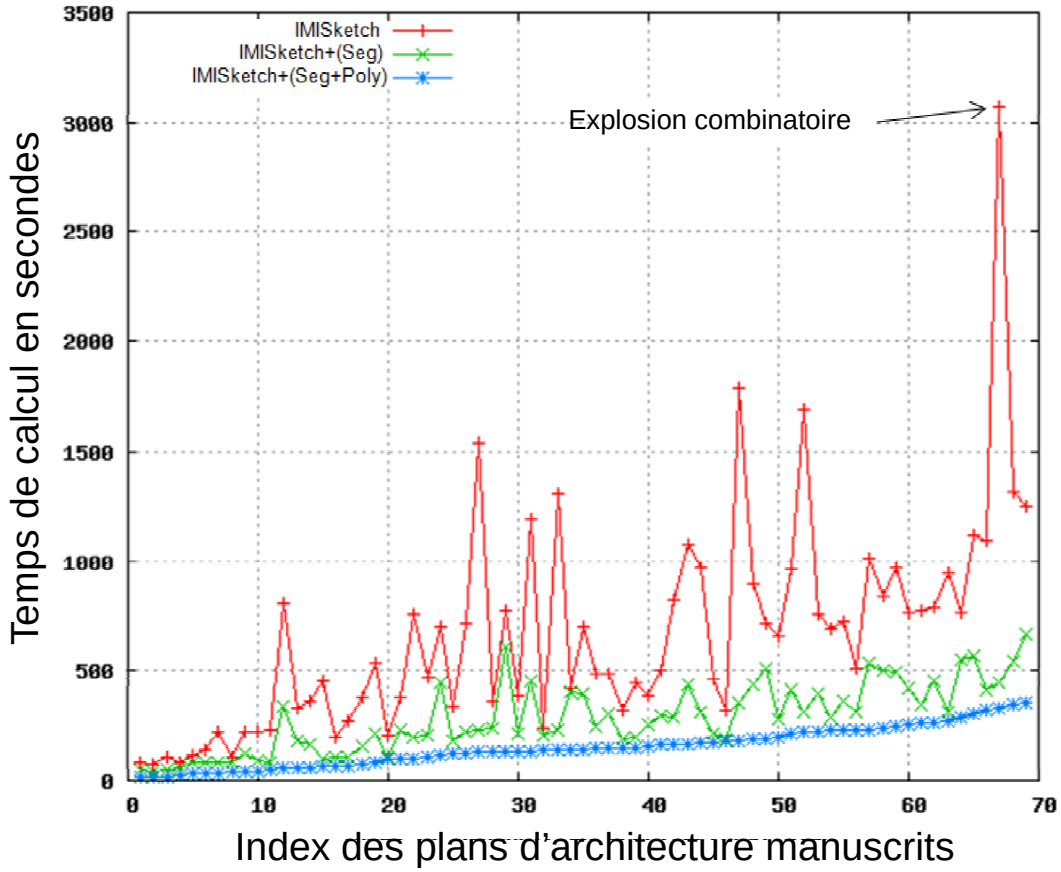


Figure 6.7 – Le temps de calcul par image montrant la présence d'explosion combinatoire avec *IMISketch* (en rouge) et leur disparition avec *IMISketch+*

Les détails de temps de calcul par image des trois versions (IMISketch, IMISketch+(Seg) et IMISketch+(Seg&poly)) sont présentés dans la figure 6.7. Les images sont ordonnées selon le temps de calcul en fonction de IMISketch+(Seg&poly)). Nous passons de 11 minutes de temps de calcul moyen par image (avec IMISketch classique) à 4 minutes 43 secondes (avec IMISketch+(Seg)), soit un gain de 57%. IMISketch+(Seg&poly) réduit encore le temps de calcul pour atteindre 2 minutes 31 se-

condes, ce qui présente un gain total de 77%. Selon la complexité des plans, le gain de temps de calcul peut atteindre 90% par image (l'image avec un indice 67). Notons que, dans la figure 6.7, les pics de temps de calcul (liés aux explosions combinatoires) ont complètement disparu avec la méthode optimisée, ce qui limite le risque d'avoir une longue attente pour interpréter un plan complexe. Les temps de calcul sont obtenus dans des conditions réelles, c'est-à-dire en présence d'un utilisateur dans la boucle d'analyse. Aujourd'hui, le temps de calcul moyen est de 2 minutes 31 par image.

Les résultats expérimentaux sont très encourageants. Ils suggèrent qu'il est possible d'introduire une analyse en largeur en évitant l'explosion combinatoire. Cela conforte l'idée de concevoir un système interactif pour la reconnaissance de documents. La sollicitation utilisateur, grâce à l'analyseur, garantit l'obtention des taux de reconnaissance très élevé, même dans le cas de documents complexes. L'utilisation de la primitive polygone n'a pas d'impact négatif sur le taux de reconnaissance structurelle, en outre, elle réduit le nombre d'interventions de l'utilisateur lors de l'analyse et accélère également le temps de calcul.

6.5 Test unitaire : méthode basée sur une exploration hybride : largeur - profondeur

Dans sa version le plus aboutie, notre méthode IMISketch adopte une exploration hybride pour la construction des arbres d'analyse. Cette stratégie permet de bien gérer l'incertitude en se basant sur l'exploration en largeur, tout en exploitant la réduction de la combinatoire offerte par l'exploration en profondeur. L'objectif de cette stratégie est de réduire le temps d'analyse du document afin de répondre aux critères d'acceptabilité du système de rétroconversion interactif. Pour bien montrer l'amélioration du système en utilisant l'exploration hybride, nous allons le tester sur 7 plans de *la base complexe*.

La première expérience consiste à montrer les améliorations en temps de calcul de notre approche hybride nommée *IMISketch hybride+(Seg&poly)*, par rapport à la version optimisée nommée *IMISketch+(Seg&poly)* (tableau 6.5). Pour cela, nous lançons deux tests sur les mêmes plans d'architecture. Le premier test est fait par l'analyseur *IMISketch hybride+(Seg&poly)* et le deuxième par la méthode *IMISketch+(Seg&poly)*. Nous comparons le nombre de nœuds créés dans l'arbre d'analyse. L'interprétation d'un plan par l'analyseur *IMISketch+(Seg&poly)* nécessite la création de plus de 5300 nœuds par plan et donc un temps de calcul très important qui ne permet pas à l'analyse d'aboutir. L'interprétation des mêmes plans avec *IMISketch hybride+(Seg&poly)* nécessite une moyenne de 1713 nœuds par plan, ce qui représente un gain de plus

de 70%. Cette nouvelle approche d'analyse hybride permet donc d'avoir une méthode interactive, avec un temps de calcul raisonnable pour l'utilisateur.

Version	Type de primitive	Construction des arbres d'analyse
IMISketch + (Seg&poly)	Segments et polygones	En utilisant les optimisations de IMISketch optimisé (section 4.3.2.2.a)
IMISketch hybride + (Seg&poly)	Segments et polygones	En utilisant les optimisations de IMISketch optimisé et et IMISketch hybride section (4.3.2.2.b)

Tableau 6.5 – Caractéristiques des deux versions d'IMISketch utilisées dans cette expérimentation

La seconde expérience porte sur l'évaluation du taux de reconnaissance structurelle, à savoir l'interprétation de primitives en murs, ouvrants et mobiliers, en utilisant la stratégie hybride, en présence de l'utilisateur dans la boucle d'analyse. Le processus d'analyse sollicite l'utilisateur en cas d'ambigüité en lui présentant l'ensemble des hypothèses concurrentes. L'utilisateur choisira alors la bonne hypothèse. Notre objectif est ici de vérifier que l'utilisation de l'exploration hybride ne dégrade pas les performances de reconnaissance.

Nous évaluons ce taux de reconnaissance sur les 7 plans d'architecture manuscrits (figure 6.1). Le taux de reconnaissance structurelle avec l'approche hybride atteint environ 95,60%. Les erreurs restantes (environ 4,4%) sont dues principalement soit à un mauvais calibrage de la taille du contexte local de recherche, soit à certains symboles mal dessinés. L'application de *IMISketch+(Seg&poly)* sur les mêmes plans d'architecture engendre des explosions combinatoires. En comparant avec les résultats obtenus dans la section 6.4 (un taux de reconnaissance de 97,93% sur les plans de la *base simplifiée*), nous pouvons estimer que le passage à l'exploration hybride maintien une bonne qualité de reconnaissance structurelle.

Avec la stratégie adoptée par *IMISketch hybride+(Seg&poly)*, nous pouvons faire face aux problèmes complexes où le mobilier est attaché au mur. Les figures 6.8(a) et 6.8(b) présentent des exemples de mobiliers partageant une partie avec des murs. La figure 6.8(c) illustre un exemple de meuble qui a une composante fixée au mur. Les figures 6.8(d) et 6.8(e) présentent des exemples de mobiliers collés à des ouvrants. L'interprétation de ces cas est très difficile avec la méthode *IMISketch+(Seg&poly)* et engendre une explosion combinatoire, mais ces cas complexes peuvent être résolus avec la méthode *IMISketch hybride+(Seg&poly)*.

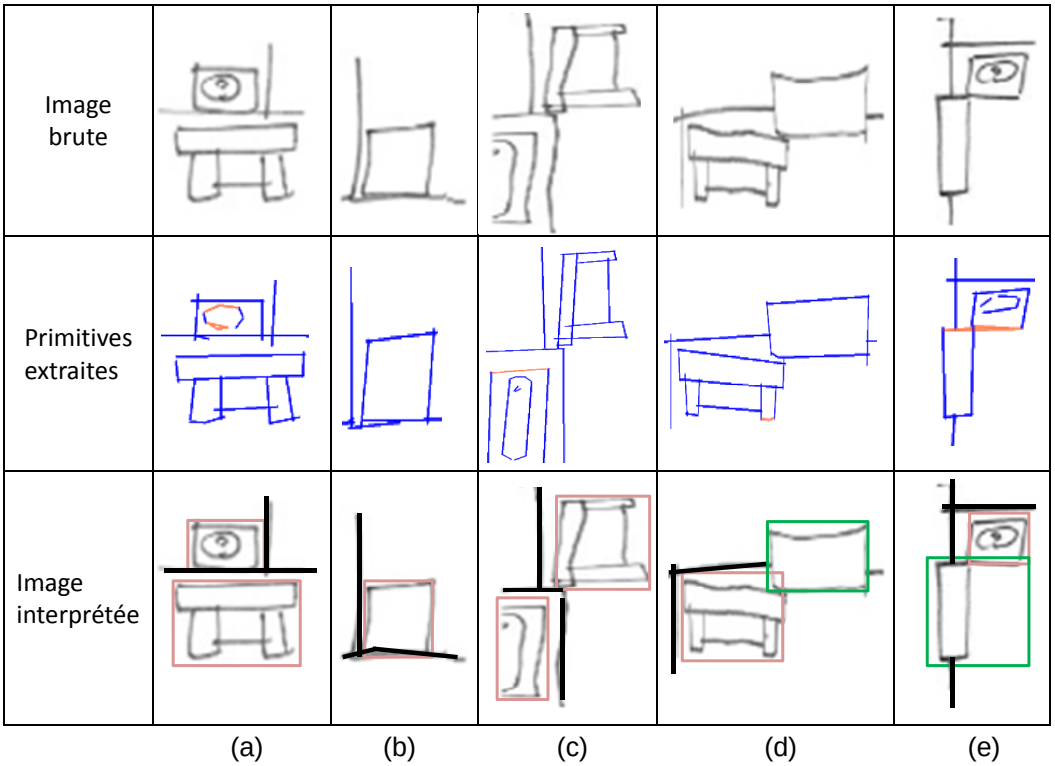


Figure 6.8 – Exemple des interprétations réussies sur les exemples complexes avec *IMISketch hybride+(Seg&poly)*. Les rectangles verts illustrent les ouvrants reconnus par l'analyseur. Les rectangles violets localisent les mobiliers. Les murs sont illustrés par des traits noirs

6.6 Évaluation globale de la méthode *IMISketch*

Les expérimentations présentées dans les parties précédentes sont des tests unitaires vérifiant l'impact des différentes contributions de la thèse. Dans cette section, nous présentons une évaluation globale de la méthode. Dans cette expérience, nous fixons la taille du contexte local de recherche selon les expériences faites précédemment (détail dans la section 5.3). Nous avons déduit de la section 6.2 le seuil d'ambigüité adéquat qui garantit le meilleur compromis reconnaissance/sollicitation. De plus, nous avons utilisés deux classifieurs (non évolutif dans cette expérimentation) pour la reconnaissance des mobiliers et des ouvrants. Ces classifieurs peuvent solliciter l'utilisateur en cas d'ambiguïté de forme, c'est-à-dire dans le cas où le classifieur considère que plusieurs étiquettes peuvent être attribuées à un symbole.

Nous avons testé notre méthode interactive sur la *base complexe* (plus de détails dans la section 6.1). Nous avons divisé cette base en une base d'apprentissage composée de 9 plans d'architecture pour initialiser le classifieur et une base de test composée de 15 plans d'architecture avec une moyenne de 84 symboles (murs, ouvrants et mobiliers)

par plan. L'étape d'extraction de primitives donne une moyenne de 302 primitives (segments et polygones) par plan.

Nombre de plan d'architecture	15
Nombre de symboles	961
Taux de reconnaissance de plan	93.4%
Nombre moyenne de sollicitations structurelles par plan	5
Pourcentage de sollicitations structurelles utiles	25%
Nombre moyenne de sollicitation classifieur par plan	5
Pourcentage de sollicitations du classifieur utiles	49%

Tableau 6.6 – Taux de reconnaissance des plans d'architecture

L'analyse totale des 15 plans d'architecture montre que le taux de reconnaissance total atteint 93,4%. Le tiers des erreurs obtenues est lié à l'étiquetage du classifieur, c'est-à-dire que le symbole est bien reconnu structurellement mais mal reconnu par le classifieur (mal étiqueté). Les erreurs restantes sont liées à des erreurs de reconnaissance structurelle dues essentiellement à des symboles mal dessinés. L'utilisateur intervient en moyenne 5 fois par plan d'architecture pour résoudre des ambiguütés structurelles, et dans 25% des cas l'utilisateur ne valide pas la meilleure hypothèse trouvée par le système (l'hypothèse ayant le meilleur score). L'analyseur sollicite l'utilisateur dans uniquement 1.6% des interprétations validées.

Dans la section 6.2, nous avons constaté, avec les mêmes conditions (seuil d'ambiguïté, taille de contexte local de recherche) mais sans l'intégration du classifieur, que l'utilisateur était sollicité 12 fois en moyenne par plan (dont 49% d'interventions utiles). Cette comparaison, avec le chiffre des interventions, montre que le classifieur permet de minimiser des éventuels cas d'ambiguïtés structurelles grâce au poids qu'il donne à l'hypothèse et donc de diminuer le nombre d'intervention utilisateur.

Le classifeur sollicite également l'utilisateur s'il n'est pas sûr de l'étiquette qu'il a donné au symbole. Le test fait sur les 15 plans d'architecture donne une moyenne de 5 sollicitation par plan dont 50% sont des interventions utiles, c'est-à-dire que l'utilisateur ne choisit pas l'étiquette proposée par le classifieur. Le tableau 6.6 résume les résultats obtenus.

Un exemple d'interprétation de plan d'architecture est illustré dans la figure 6.9, décrivant le plan d'architecture à interpréter, la figure 6.10 présentant l'ensemble de

primitive extrait du plan d'architecture et la figure 6.11 illustrant le résultat final de l'interprétation.

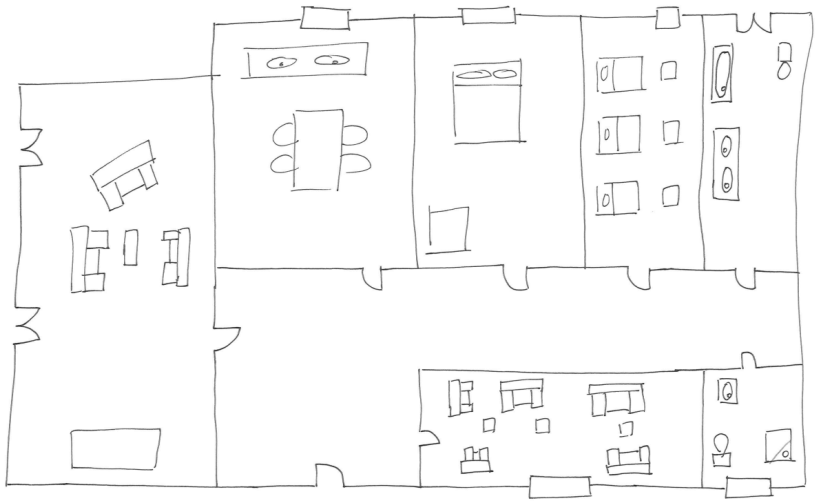


Figure 6.9 – Exemple de plan d'architecture à interpréter

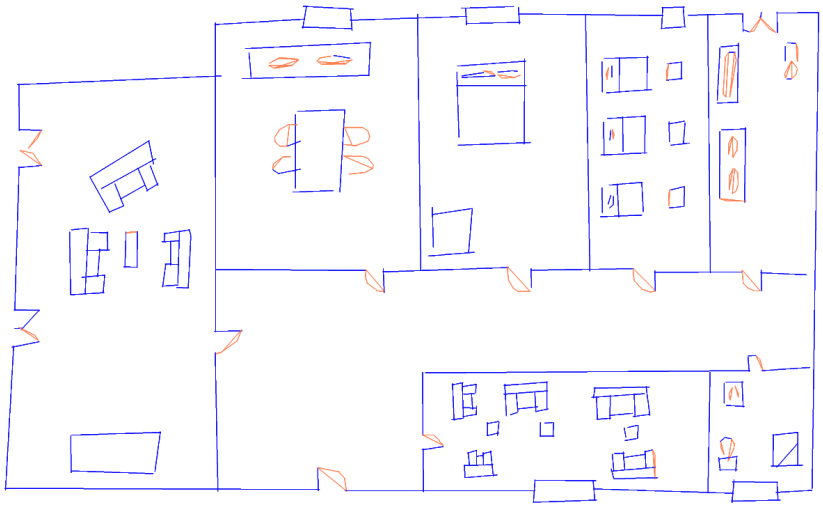


Figure 6.10 – Primitives extraites du plan d'architecture décrit dans la figure 6.9

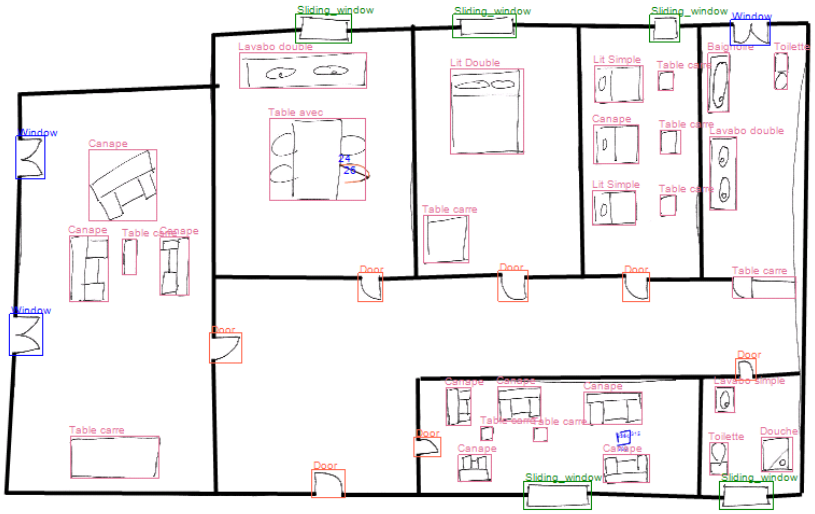


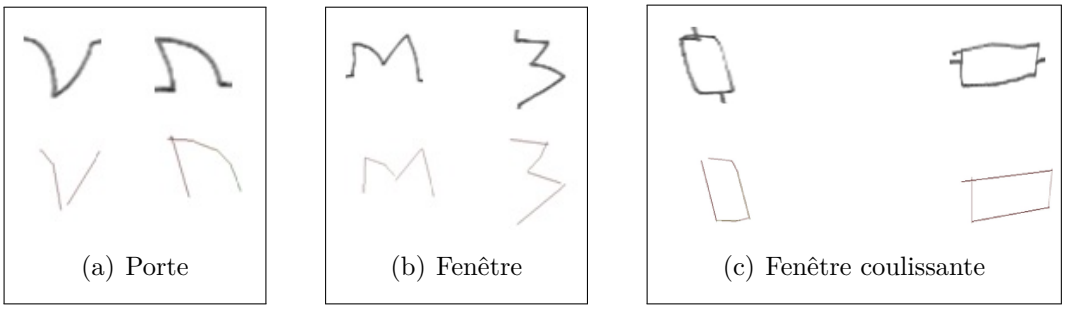
Figure 6.11 – Plan d'architecture interprété

6.7 Test unitaire : apport de l'utilisation d'un classifieur incrémental

Dans les travaux à venir, la version incrémentale du classifieur sera intégrée pour la reconnaissance des ouvrants et des mobiliers. Nous reportons, dans cette section, les résultats d'un test unitaire sur la reconnaissance d'ouvrant.

Nous analysons dans cette partie l'impact de l'apprentissage incrémental sur l'évolution des performances en classification. Cette performance est mesurée par deux taux : le taux d'erreur et le taux de rejet. Notre objectif est de réduire les erreurs de reconnaissance et de minimiser autant que possible le nombre d'interventions des utilisateurs.

La base de données utilisée dans ces expériences contient 1500 exemples de symboles de trois classes différentes (porte, fenêtre, fenêtre coulissante), extraits de *la base simplifiée*. Quelques exemples de symboles sont présentés dans la figure 6.12.



(a) Porte (b) Fenêtre (c) Fenêtre coulissante

Figure 6.12 – Exemples de symboles traités par le classifieur incrémental

Nous divisons l'ensemble des données en trois sous-ensembles :

- Sous-ensemble d'apprentissage initial : utilisé pour entrainer le classifieur de façon supervisée, c'est-à-dire que chaque étiquette d'un symbole est donnée par l'utilisateur. Ce sous-ensemble contient 113 exemples.
- Sous-ensemble d'évaluation : utilisé pour évaluer la performance du classifieur par la mesure de l'erreur et du rejet. 378 exemples sont utilisés dans ce sous-ensemblee.
- Sous-ensemble d'apprentissage incrémental : utilisé pour améliorer la performance du classifieur en sollicitant l'utilisateur quand une confusion de rejet est détectée. Il contient 1019 exemples.

Les expériences ont été répétées pour des seuils de rejet différents. La figure 6.13 présente des points de performance différents (erreur, rejet) en utilisant seulement le sous-ensemble d'apprentissage initial dans la première courbe (en bleu), et puis en utilisant le sous-ensemble d'apprentissage progressif, dans la deuxième courbe (en rouge). Notons que le processus d'apprentissage progressif améliore les performances du classifieur grâce à la sollicitation de l'utilisateur pour les exemples rejetés. Nous constatons que l'intégration de l'apprentissage incrémental dans le processus d'analyse améliore les performances de la reconnaissance graphique des symboles et donc les performances globales du système.

Afin de donner un ordre d'idée sur le nombre d'interventions requises par le processus d'apprentissage progressif, nous montrons l'évolution du taux d'erreur (la figure 6.14) et du taux de rejet (la figure 6.15) selon le nombre d'interventions, pour une valeur du seuil de rejet bien déterminée $\lambda = 0.5$. Cette expérience permet de trouver le bon compromis entre le taux de reconnaissance et le nombre d'interventions. Nous remarquons que le taux d'erreur estimé du classifieur a été réduit d'environ 30% après 40 sollicitations utilisateur, et que le taux de rejet estimé a également été réduit d'environ 22%.

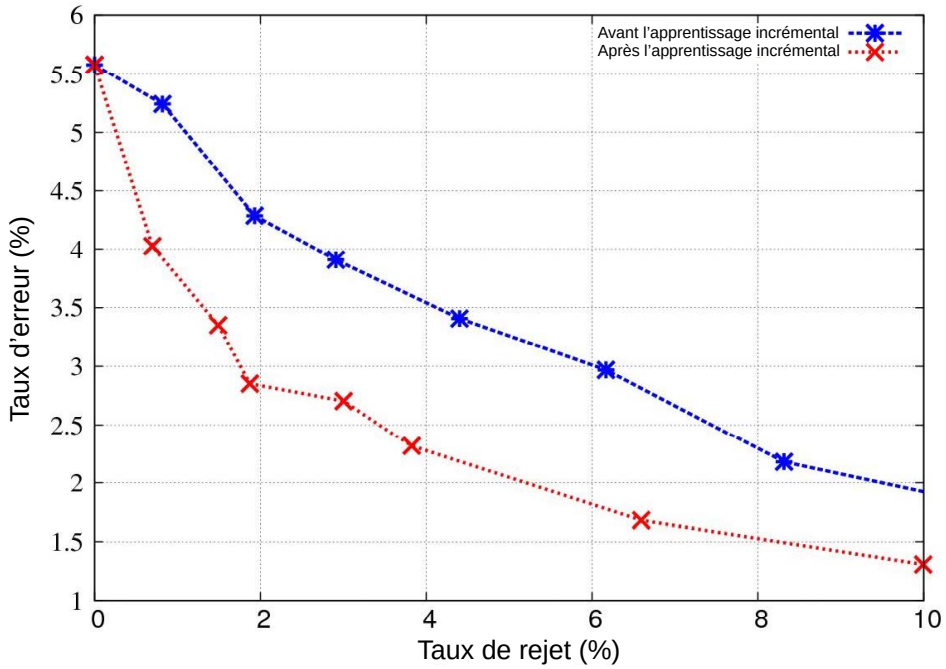


Figure 6.13 – La courbe d'erreur/rejet avant et après le processus d'apprentissage incrémental

À l'issue de cette expérience, nous avons montré que la présence d'un classifieur incrémental améliore le taux de reconnaissance des symboles.

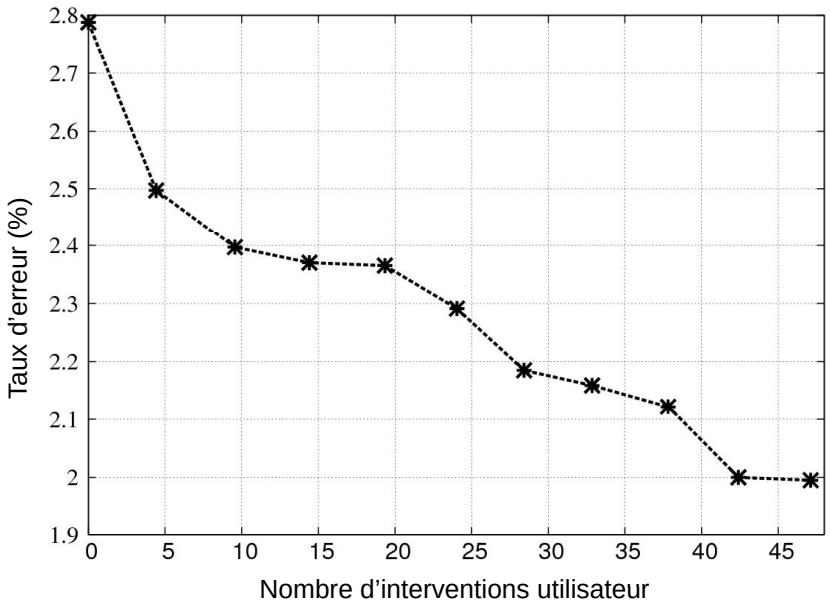


Figure 6.14 – Réduction du taux d'erreur durant le processus d'apprentissage incrémental ($\lambda = 0.5$)

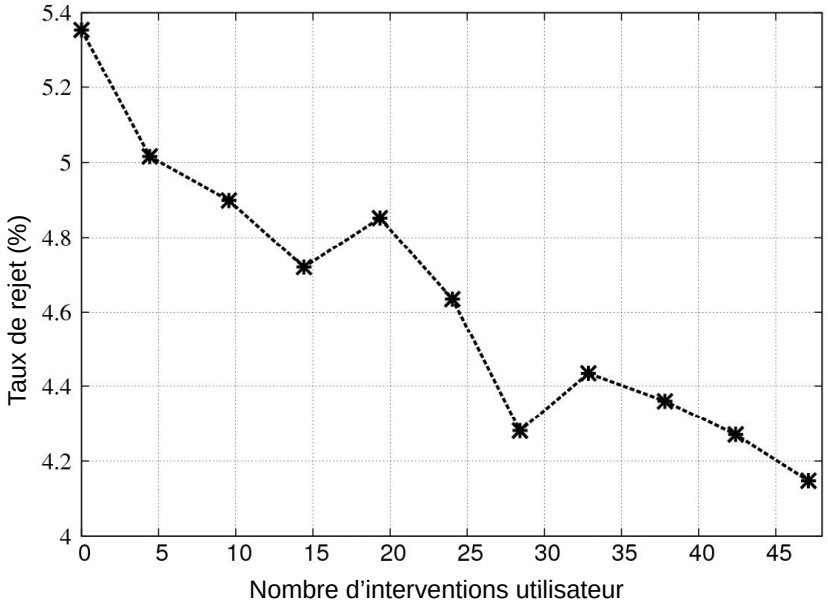


Figure 6.15 – Réduction du taux de rejet durant le processus d'apprentissage incrémental ($\lambda = 0.5$)

Une troisième expérience permet d'évaluer la capacité d'adaptation du classifieur en ajoutant de nouvelles classes durant la phase d'apprentissage. Pour cette expérience, nous utilisons :

- un classifieur qui contient deux classes (portes et fenêtres), initialisé par trois symboles par classe du sous-ensemble d'apprentissage initial.
- une base d'apprentissage incrémental, prise du sous-ensemble d'apprentissage incrémental, qui contient 10 portes, 10 fenêtres et 10 fenêtres coulissantes.
- une base de données d'évaluation (BE), prise du sous-ensemble d'évaluation, qui contient 118 portes, 99 fenêtres et 14 fenêtres coulissantes.

Cette expérience comporte deux parties. La figure 6.16 détaille le déroulement de cette expérience. La première a pour but d'améliorer la reconnaissance des portes et des fenêtres. Dans la phase d'apprentissage de cette partie, le classifieur apprend à chaque cycle une porte et une fenêtre. Chaque phase d'apprentissage est suivie par une phase d'évaluation permettant de calculer le taux d'erreurs de reconnaissance de la base d'évaluation. La figure 6.17 décrit l'évaluation du classifieur durant cette partie. Le taux d'erreurs diminue de 15,1% à 8,7%.

La deuxième partie consiste à enrichir le classifieur, dans la phase d'apprentissage par une nouvelle classe (fenêtre coulissante). En fait, le classifieur apprend, à chaque cycle, une porte, une fenêtre et deux fenêtres coulissantes. Après quelques cycles d'ap-

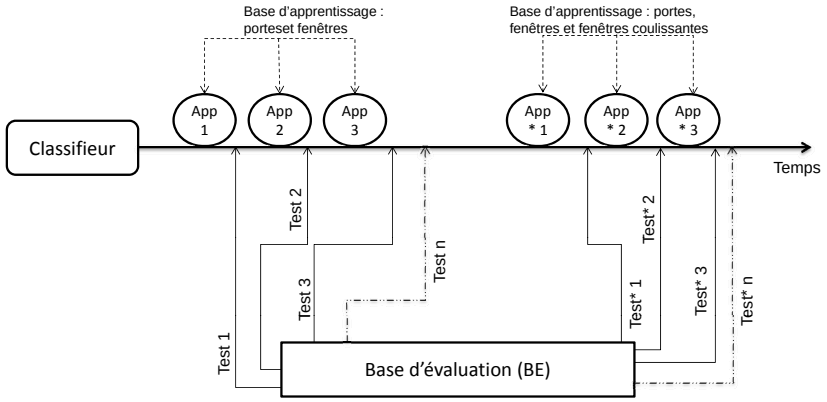


Figure 6.16 – Déroulement de la troisième expérience

prentissage, le classifieur trouve sa stabilité et absorbe cette classe. Pendant cette partie, le taux d'erreurs décroit de 18,2% à 3,9% (figure 6.17).

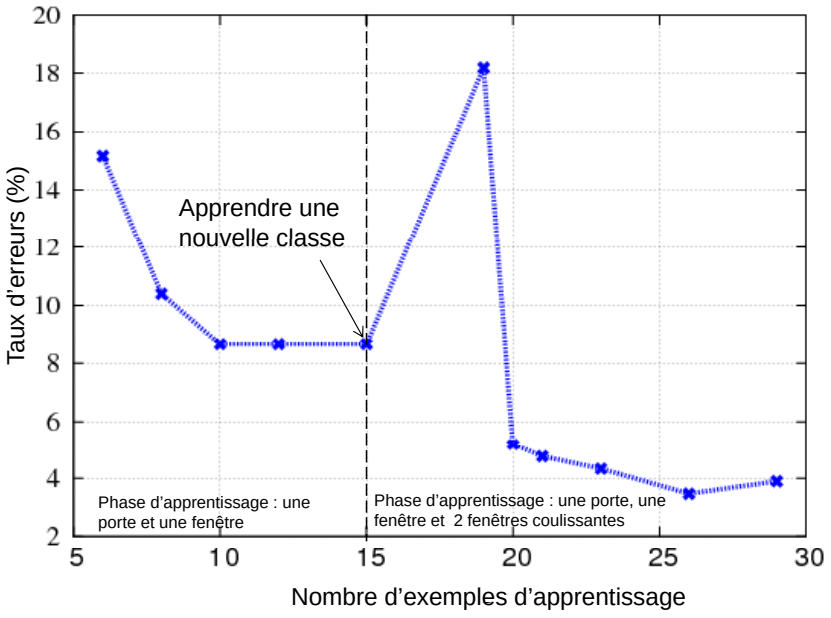


Figure 6.17 – Évolution de la performance pendant la phase d'apprentissage (intégration d'une nouvelle classe)

Nous avons montré, dans ce test unitaire, la performance d'un classifieur basé sur l'apprentissage incrémental et sa capacité à intégrer de nouvelles classes au cours de la phase d'analyse des documents structurés. La prochaine étape consistera à intégrer ce classifieur incrémental dans la méthode IMISketch.

6.8 Conclusion

Nous avons validé dans ce chapitre les principales caractéristiques de la méthode interactive IMISketch. Cette méthode a été appliquée sur deux ensembles de plans d'architecture dessinés à main levée.

Nous avons divisé les expérimentations en deux parties. La première partie a pour but de valider chaque contribution de la thèse d'une manière séparée. Nous avons commencé par décrire l'impact d'intégrer un utilisateur dans la boucle d'analyse sur le taux de reconnaissance, ce qui limitera la phase de vérification/correction a posteriori qui peut être fastidieuse.

Nous avons montré également les optimisations liées à l'exploration hybride pour la réduction de la combinatoire. Cette exploration garantit également la gestion d'incertitude liée essentiellement à la nature manuscrite du document en mettant en compétition des hypothèses concurrentes.

Les différentes contributions ont été collectées dans une expérimentation d'évaluation globale de notre de système d'interprétation interactive. Un autre test unitaire permettant d'avoir un système auto-évolutif grâce des classifieurs incrémentaux a été validé. L'intégration de cette contribution dans notre méthode *IMISketch* fera l'objet de nos travaux à venir.

Cinquième partie

Conclusion et perspectives

CHAPITRE 7

Conclusion et perspectives

7.1 Conclusion

Cette thèse entre dans le cadre du projet ANR-Mobisketch. Ce projet vise à élaborer une solution logicielle générique orientée stylo pour la réalisation de documents techniques : schémas, plans... L'objectif est d'aboutir à un continuum entre un document technique sous sa forme papier et ce même document sous sa forme numérique interprétée. Le principe est de reconnaitre un document existant numérisé ou vectoriel pour en extraire une représentation numérique interprétée et pouvoir ensuite l'éditer, en contexte de mobilité terrain, à travers une modalité d'interaction «homme-document» orientée stylo.

Ce continuum nécessite deux analyseurs cohérents : un pour la phase de rétroconversion et un autre pour la composition/édition. Nous nous sommes intéressés dans cette thèse à l'analyseur pour la rétroconversion. Le but de nos travaux était d'élaborer une approche interactive, générique et incrémentale pour la rétroconversion des documents structurés. La rétroconversion consiste à interpréter un document a posteriori. L'originalité de notre méthode est la sollicitation de l'utilisateur durant la phase de rétroconversion.

Nous avons conçu la méthode IMISketch (acronyme pour *Interactive Method for Interpretation of Sketch*) permettant de réaliser un système interactif pour la rétroconversion des documents structurés. Notre méthode offre l'avantage d'être générique, et donc de pouvoir être utilisée avec des documents de natures variées. Elle est caractérisée par :

- une utilisation de la théorie des langages visuels, afin de modéliser différentes natures de documents, leur structure et leur symboles ;
- une interaction homme-document, permettant de prendre en compte l'utilisateur de manière originale pour en faire un acteur du processus d'interprétation ;

- une exploration hybride (largeur/profondeur) pour la construction des arbres d'analyse ;
- une possibilité d'évoluer le système par un mécanisme d'interprétation incrémentale capable d'ajouter de nouvelles classes de symbole durant le processus d'analyse.

La première contribution de la méthode IMISketch est l'intégration de l'utilisateur durant la phase d'interprétation. Le système de reconnaissance est capable de solliciter l'utilisateur dans les cas d'ambiguïté. Deux types d'ambiguïté peuvent se présenter :

- une ambiguïté structurelle : si le processus de décision hésite entre deux ou plusieurs segmentations possibles (par exemple entre une fenêtre et deux portes) ;
- une ambiguïté de forme : si le processus de décision hésite entre deux ou plusieurs classes possibles pour le même objet reconnu structurellement. Par exemple, pour un élément reconnu structurellement en un ouvrant si le processus de décision hésite entre une fenêtre et une porte.

La sollicitation de l'utilisateur durant l'analyse évite la propagation d'erreur et donc une phase de vérification/correction a posteriori qui pourrait être fastidieuse.

La deuxième contribution de la méthode IMISketch consiste à réduire la combinatoire engendrée par le système de rétroconversion en largeur, nécessaire pour mettre les hypothèses en concurrence. Plusieurs axes ont été abordés pour résoudre ce problème :

- Définition du contexte local de recherche : nous limitons l'exploration des arbres d'analyse dans le contexte local de recherche qui est une zone du document permettant d'avoir de l'information nécessaire pour bien interpréter un élément.
- Double représentation des primitives : nous avons utilisé des segments et des polygones, chaque polygone étant formé d'un ensemble de segments. La représentation «grossière» est utilisée principalement pour la reconnaissance structurelle. Cette représentation réduit la combinatoire en utilisant les primitives segments et polygones en entier. Dans le cas des plans d'architecture, ces symboles sont les ouvrants, les mobiliers et les murs. La représentation plus fine (utilisant tous les sous-segments du polygone) garantit une meilleur reconnaissance des symboles isolés structurellement. Cette représentation permet de reconnaitre les symboles en envoyant au classifieur les segments et les segments formant les polygones.
- Exploration hybride (largeur/profondeur) permettant de mettre en compétition les hypothèses concurrentes et d'éviter de construire des nœuds inutiles.

La troisième contribution est un enrichissement des grammaires GMC-PC. L'originalité consiste à piloter le type d'exploration des arbres d'analyse à partir du formalisme grammatical. L'application de la règle de production indique si elle peut être concurrente avec d'autres hypothèses ou non. Dans notre modélisation, nous considérons que

la concurrence aura lieu uniquement entre les objets complets (les objets obtenus dans le résultat final de l'interprétation).

Toutes ces contributions ont été validées sur des croquis de plans d'architecture de complexités différentes. Afin de mettre en valeur l'apport de chaque contribution, nous avons fait des tests unitaires pour valider chaque contribution. Les résultats ainsi obtenus sont encourageants. Ensuite, nous avons rassemblé toutes ces contributions pour une évaluation globale de notre méthode interactive IMISketch. Aujourd'hui, nous avons finalisé la phase de prototypage. Le projet ANR se termine en mai 2013. Les derniers travaux concerneront le développement d'un démonstrateur, la réalisation de tests utilisateurs sur ce démonstrateur et l'étude des possibilités de transfert industriel.

7.2 Contributions scientifiques et publications

La méthode IMISketch et ses différentes évolutions ont fait l'objet de plusieurs publications scientifiques, à la fois dans la communauté de la reconnaissance de formes [GALA11, GALA12] et dans celle de l'analyse des documents structurés [GAL12a, GAL12b, GLA12, GMLA11]. Elle a également fait l'objet d'une soumission dans un journal international «Pattern Recognition Letters [1]».

Dans [GMLA11], nous avons décrit les différentes parties de la notre méthode IMISketch afin de mettre en valeur son interactivité. Nous avons présenté le moteur d'analyse dans sa globalité et son mode de fonctionnement afin d'engendrer des hypothèses concurrentes. Ces hypothèses peuvent élever des ambiguïtés qui seront traitées par des sollicitations utilisateur. Nous avons montré également que le moteur d'analyse est générique et ne dépend pas du domaine d'application.

Nous avons également publié sur les optimisations afin d'éviter les explosions combinatoires et réduire les temps de calcul. Dans [GAL12a, GAL12b], nous nous sommes intéressés aux optimisations en termes de contraintes structurelles pour l'exploration en largeur des arbres d'analyse (section 4.3.2.2.a). L'article soumis à «Pattern Recognition Letters» présente en détail le choix des primitives et la description des différentes parties. Il montre également le gain en temps de calcul obtenu à partir des optimisations en terme de contraintes structurelles et la double représentation des primitives. Toutes ces publications ont porté sur des plans d'architecture contenant uniquement des murs et des ouvrants (portes, fenêtres, fenêtres coulissantes).

En passant à des plans d'architecture plus complexes, contenant des murs, des ouvrant et des mobiliers, nous avons remarqué que les optimisations déjà faites ne

1. Papier en révision mineure

suffisaient pas pour répondre à nos exigences et à nos besoins. Pour cela, nous avons proposé une construction hybride des arbres d'analyse publiées dans [GLA12].

En terme de reconnaissance de forme, nous avons validé l'apprentissage incrémental dans la méthode IMISketch. L'intégration d'un classifieur incrémental permet d'ajouter de nouvelles classes de symboles durant l'analyse. Ces travaux ont fait l'objet d'une publication dans une conférence internationale [GALA11], étendue à un chapitre de livre LNCS [GALA12].

7.3 Perspectives

Les travaux que nous avons réalisés au cours de cette thèse peuvent avoir différentes perspectives.

La première perspective consiste à intègrer le classifieur incrémental dans la méthode IMISketch. Le test unitaire sur l'apport de cette intégration donne des résultats encourageants. La possibilité d'ajouter de nouvelles classes de symboles, offerte par le classifieur, permettra de passer à une méthode auto-évolutive capable d'enrichir sa connaissances a priori à partir de la sollicitation utilisateur.

D'autres travaux à court terme concerneront l'extension des domaines d'application de l'approche IMISketch. La méthode IMISketch est capable d'interpréter des documents de différentes natures (manuscrit, imprimé, vectoriel) et de différents domaines (plan d'architecture, schéma électrique, etc.). Nous allons travailler sur l'application de notre méthode sur d'autres types de documents et d'autres domaines d'application. Cela permettra de mettre à l'épreuve le caractère générique de l'approche.

Un autre axe de recherche consiste à explorer les possibilités de laisser à l'utilisateur la capacité de mettre en pause le processus d'analyse pour effectuer manuellement une correction ou une interprétation. Cette intervention permettra de corriger les erreurs d'interprétation (non détectées par l'analyseur) au plus tôt, et ainsi d'éviter une propagation d'erreurs en chaîne.

Une perspective à plus long terme qui est défi scientifique, concerne l'apprentissage automatique de la formalisation. En effet, décrire la connaissance modélisant un document structuré à l'aide d'une grammaire est une tâche assez fastidieuse. Bien que ce soit plus rapide, plus sûr et certainement moins contraignant que d'écrire le code source complet correspondant, ce processus reste long. Le but serait alors de faciliter l'écriture des règles de grammaire en apprenant automatiquement, à partir d'exemples, autant d'informations que possible. Il n'est pas aujourd'hui envisageable d'apprendre l'ensemble des productions grammaticales qui entrent en jeu ; cependant, il serait in-

téressant de définir de nouveaux procédés pour assister le concepteur, par exemple en développant une interface pour spécifier les contraintes sans avoir à écrire directement les productions grammaticales.

Références de l'auteur

[GAL12a] Achraf Ghorbel, Eric Anquetil, and Aurélie Lemaitre. Optimisation de l'analyse en largeur pour une approche structurelle d'interprétation de croquis. In *Congrés Francophone sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA'12)*, 2012.

[GAL12b] Achraf Ghorbel, Eric Anquetil, and Aurélie Lemaitre. Optimization analysis based on a breadth-first exploration for a structural approach of sketches interpretation. In *10th IAPR International Workshop on Document Analysis Systems (DAS'12)*, pages 240–244, 2012.

[GALA11] Achraf Ghorbel, Abdullah Almaksour, Aurélie Lemaitre, and Eric Anquetil. Incremental learning for interactive sketch recognition. In *Nineth IAPR International Workshop on Graphics RECognition (GREC 2011)*, pages 93–96, 2011.

[GALA12] Achraf Ghorbel, Abdullah Almaksour, Aurélie Lemaitre, and Eric Anquetil. Incremental learing for interactive sketch recognition. volume LNCS 7423. Springer Berlin / Heidelberg, 2012.

[GLA12] Achraf Ghorbel, Aurélie Lemaitre, and Eric Anquetil. Competitive hybrid exploration for off-line sketches structure recognition. In *Proceedings of International Conference on Frontiers of Handwriting Recognition (ICFHR'2012)*, 2012.

[GMLA11] Achraf Ghorbel, Sébastien Macé, Aurélie Lemaitre, and Eric Anquetil. Interactive competitive breadth-first exploration for sketch interpretation. In *International Conference on Document Analysis and Recognition (ICDAR'2011)*, pages 1195–1199, 2011.

Bibliographie

[1] Map : Multi-angled parallelism for feature extraction from topographical maps. *Pattern Recognition*, 24(6) :479 – 488, 1991. 29

[2] S. Ablameyko. An introduction to interpretation of graphic images. SPIE-International Society for Optical Engineering, 1997. 34

[3] S. Ablameyko and T. Pridmore. Machine interpretation of line drawing images technical drawings, maps and diagrams. *Recherche*, 67 :02, 2000. 33

[4] S. Adam, JM Ogier, C. Cariou, R. Mullot, J. Gardes, and Y. Lecourtier. Utilisation de la transformée de fourier-mellin pour la reconnaissance de formes multi-orientées et multi-échelles : application à l'analyse automatique de documents techniques. *Traitement du signal*, 18(1) :17, 2001. 37, 38

[5] C. Ah-Soon. A constraint network for symbol detection in architectural drawings. *Graphics Recognition Algorithms and Systems*, pages 80–90, 1998. 21

[6] C. Ah-Soon and K. Tombre. Architectural symbol recognition using a network of constraints. *Pattern Recognition Letters*, 22(2) :231–248, 2001. 40

[7] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Improved automatic analysis of architectural floor plans. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 864–869. IEEE, 2011. 22

[8] A. Almaksour and E. Anquetil. Improving premise structure in evolving takagi–sugeno neuro-fuzzy classifiers. *Evolving Systems*, 2(1) :25–33, 2011. 78, 126, 128

[9] C. Alvarado and R. Davis. Sketchread : a multi-domain sketch recognition engine. In *ACM SIGGRAPH 2007 courses*, page 34. ACM, 2007. 23

[10] E. Anquetil. *Modélisation et reconnaissance par la logique floue : application à la lecture automatique en-ligne de l'écriture manuscrite omni-scripteur*. PhD thesis, 1997. 66

[11] J. Anstice, T. Bell, A. Cockburn, and M. Setchell. The design of a pen-based musical input system. In *Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on*, pages 260–267. IEEE, 1996. 26

[12] Y. Aoki, A. Shio, H. Arai, and K. Odaka. A prototype system for interpreting hand-sketched floor plans. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 747–751. IEEE, 1996. 21

[13] R. Arandjelović and T.M. Sezgin. Sketch recognition by fusion of temporal and image-based features. *Pattern Recognition*, 44(6) :1225–1234, 2011. 38

[14] Juan F. Arias, Chan P. Lai, Surekha Surya, Rangachar Kasturi, and Atul Chhabra. Interpretation of telephone system manhole drawings. *Pattern Recogn. Lett.*, 16(4) :355–369, April 1995. 22

[15] J.L. Arnott, A.F. Newell, and N. Alm. Prediction and conversational momentum in an augmentative communication system. *Communications of the ACM*, 35(5) :46–57, 1992. 53

[16] Ahmad-Montaser Awal. *Reconnaissance de structures bidimensionnelles : Application aux expressions mathématiques manuscrites en-ligne*. PhD thesis, Université de Nantes, November 2010. 22

[17] C. Baber and K. S. Hone. Modeling error recovery and repair in automatic speech recognition. *Int. J. Man-Mach. Stud.*, 39(3) :495–515, September 1993. 51

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3) :346–359, 2008. 22

[19] A. Belaïd and Y. Belaïd. *Reconnaissance des formes : méthodes et applications*. InterEditions, 1992. 46

[20] A. Bhat and T. Hammond. Using entropy to distinguish shape versus text in hand-drawn diagrams. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1395–1400. Morgan Kaufmann Publishers Inc., 2009. 29

[21] S. Bhattacharjee and G. Monagan. Recognition of cartographic symbols. In *in Proceedings of the IAPR Workshop on Machine Vision Applications*. Citeseer, 1994. 37

[22] Christopher M. Bishop, Markus Svensen, and Geoffrey E. Hinton. Distinguishing text from graphics in on-line handwritten ink. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, IWFHR '04, pages 142–147, Washington, DC, USA, 2004. IEEE Computer Society. 29

[23] Rachel Blagojevic, Beryl Plimmer, John C. Grundy, and Yong Wang. Using data mining for digital ink recognition : Dividing text and shapes in sketched diagrams. *Computers & Graphics*, 35(5) :976–991, 2011. 29

[24] I. Bloch. Fuzzy relative position between objects in image processing : a morphological approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(7) :657–664, 1999. 64

[25] D. Blostein and L. Haken. Using diagram generation software to improve diagram recognition : A case study of music notation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(11) :1121–1136, 1999. 51

[26] B. Bouchon-Meunier and C. Marsala. Logique floue, principes, aide à la décision. *Hermès-Lavoisier*, 2003. 66

[27] Walter S. Brainerd. Tree generating regular systems. *Information and Control*, 14(2) :217 – 231, 1969. 40

[28] S.E. Brennan and E.A. Hulteen. Interaction and feedback in a spoken language system : A theoretical framework. *Knowledge-Based Systems*, 8(2) :143–151, 1995. 51

[29] H. Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6) :574–582, 1982. 22, 40

[30] H. Bunke. Recent developments in graph matching. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 117–124. IEEE, 2000. 40

[31] H. Bunke and A. Sanfeliu. *Syntactic and structural pattern recognition : theory and applications*, volume 7. World Scientific Pub Co Inc, 1990. 42

[32] Horst Bunke and B. Haller. A parser for context free plex grammars. In *Proceedings of the 15th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '89, pages 136–150, London, UK, UK, 1990. Springer-Verlag. 49

[33] R. Cao and C.L. Tan. A model of stroke extraction from chinese character images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 368–371. IEEE, 2000. 33

[34] K.F. Chan and D.Y. Yeung. Pencalc : A novel application of on-line mathematical expression recognition technology. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 774–778. IEEE, 2001. 22

[35] Hung-Hsin Chang and Hong Yan. Vectorization of hand-drawn image using piecewise cubic bézier curves fitting. *Pattern Recognition*, 31(11) :1747 – 1755, 1998. 73

[36] Qi Chen, John Grundy, and John Hosking. An e-whiteboard application to support early design-stage sketching of uml diagrams. In *In Proceedings of the 2003 IEEE Conference on Human-Centric Computing*, pages 219–226. IEEE CS Press, 2003. 26

[37] Y.S. Chen. Segmentation and association among lines and junctions for a line image. *Pattern recognition*, 27(9) :1135–1157, 1994. 33

[38] HD Cheng, CH Chen, HH Chiu, and H. Xu. Fuzzy homogeneity approach to multilevel thresholding. *Image Processing, IEEE Transactions on*, 7(7) :1084–1086, 1998. 30

[39] HD Cheng, J.R. Chen, and J. Li. Threshold selection based on fuzzy c-partition entropy approach. *Pattern Recognition*, 31(7) :857–870, 1998. 30

[40] CK Chow and T. Kaneko. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and biomedical research*, 5(4) :388–410, 1972. 30

[41] G. Costagliola, V. Deufemia, and M. Risi. Sketch grammars : A formalism for describing and recognizing diagrammatic sketch languages. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 1226–1230. IEEE, 2005. 42, 43, 49

[42] G. Costagliola, S. Orefice, G. Polese, G. Tortora, and M. Tucci. Automatic parser generation for pictorial languages. In *Visual Languages, 1993., Proceedings 1993 IEEE Symposium on*, pages 306–313. IEEE, 1993. 42, 49

[43] G. Costagliola and G. Polese. Extended positional grammars. In *Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on*, pages 103–110. IEEE, 2000. 42, 43, 49

[44] Gennaro Costagliola, Andrea De Lucia, Sergio Orefice, and Genoveffa Tortor. A parsing methodology for the implementation of visual systems. *IEEE Trans. Softw. Eng.*, 23(12) :777–799, December 1997. 42

[45] Gennaro Costagliola, Vincenzo Deufemia, Giuseppe Polese, and Michele Risi. A parsing technique for sketch recognition systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, VLHCC '04, pages 19–26, Washington, DC, USA, 2004. IEEE Computer Society. 43

[46] Gennaro Costagliola, Genoveffa Tortora, Sergio Orefice, and Andrea De Lucia. Automatic generation of visual programming environments. *Computer*, 28(3) :56–66, March 1995. 42

[47] B. Couasnon and J. Camillerapp. Segmentation et reconnaissance de documents guidées par la connaissance a priori : application aux partitions musicales. 1996. 23

[48] Bertrand Coüasnon. Dmos, a generic document recognition method : Application to table structure analysis in a general and in a specific way. *IJDAR 2006*, 8(2) :111–122, 2006. 22, 34, 42, 44, 48

[49] C. Crimi, A. Guercio, G. Nota, G. Pacini, G. Tortora, and M. Tucci. Relation grammars and their application to multi-dimensional languages. *Journal of Visual Languages & Computing*, 2(4) :333–346, 1991. 42, 44

[50] Christian Heide Damm, Klaus Marius Hansen, and Michael Thomsen. Tool support for cooperative object-oriented design : gesture based modelling on an electronic whiteboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 518–525, New York, NY, USA, 2000. ACM. 26

[51] Denis de Brucq, Mounir Amara, Pierre Courtellemont, Philippe Wallon, and Claude Mesmin. A recursive estimation of parameters of straight lines and circles by an extended Kalman filtering. Application to the modeling of on-line Handwritten drawings. In *International Conference on Signals and Image Processing and Applications (SIPA'96)*, pages 213–216, 1996. 73

[52] A. Donaldson and A. Williamson. Pen-based input of uml activity diagrams for business process modelling. In *Proc HCI 2005 Workshop on Improving and Assessing Pen-based Input Techniques, Edinburgh*, volume 3, 2005. 26

[53] D. Dori and W. Liu. Sparse pixel vectorization : An algorithm and its performance evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(3) :202–215, 1999. 33

[54] P. Dosch, K. Tombre, C. Ah-Soon, and G. Masini. A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3(2) :102–116, 2000. 22, 21

[55] Philippe Dosch, Karl Tombre, Christian Ah-Soon, and Gérald Masini. A complete system for analysis of architectural drawings. *International Journal On Document Analysis and Recognition*, 3(2) :102–116, 2000. 71

[56] Ø. Due Trier, A.K. Jain, and T. Taxt. Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4) :641–662, 1996. 37

[57] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2) :94–102, 1970. 48

[58] H. Fahmy and D. Blostein. A graph grammar programming style for recognition of music notation. *Machine Vision and Applications*, 6(2) :83–99, 1993. 23, 40

[59] Guihuan Feng, Christian Viard-Gaudin, and Zhengxing Sun. On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming. *Pattern Recognition*, 42(12) :3215–3223, 2009. 23, 22

[60] J.A. Fitzgerald, F. Geiselbrechtinger, and T. Kechadi. Mathpad : A fuzzy logic-based recognition system for handwritten mathematics. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 694–698. IEEE, 2007. 22

[61] L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(6) :910–918, 1988. 29

[62] Andrew Forsberg, Mark Dieterich, and Robert Zeleznik. The music notepad. In *In Proceedings of the ACM Symposium on User Interface and Software Technology (UIST). ACM, ACM*, pages 203–210. Press, 1998. 26

[63] H. Freeman and L.S. Davis. A corner-finding algorithm for chain-coded curves. *Computers, IEEE Transactions on*, 100(3) :297–303, 1977. 37

[64] Isaac J. Freeman and Beryl Plimmer. Connector semantics for sketched diagram recognition. In *Proceedings of the eight Australasian conference on User interface - Volume 64*, AUIC '07, pages 71–78, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc. 34, 39

[65] U. Garain and B.B. Chaudhuri. Recognition of online handwritten mathematical expressions. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 34(6) :2366–2376, 2004. 22

[66] Leslie Gennari, Levent Burak Kara, Thomas F. Stahovich, and Kenji Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Comput. Graph.*, 29(4) :547–562, August 2005. 34, 51

[67] R. Genoe, J.A. Fitzgerald, T. Kechadi, et al. A purely online approach to mathematical expression recognition. 2006. 26

[68] E.P. Glinert, J.A. Jorge, and D.A. Vaida. Fuzzy adjacency languages and applications to spatial reasoning. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 1, pages 633–639. IEEE, 1996. 42, 44

[69] D. Goldberg and A. Goodisman. Stylus user interfaces for manipulating text. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 127–135. ACM, 1991. 52

[70] Eric J. Golin. Parsing visual languages with picture layout grammars. *J. Vis. Lang. Comput.*, 2(4) :371–393, December 1991. 42, 44, 49

[71] R.C. Gonzalez and M.G. Thomason. Syntactic pattern recognition : An introduction. 1978. 40

[72] M.D. Gross. The electronic cocktail napkin - a computational environment for working with design diagrams. *Design Studies*, 17(1) :53–69, 1996. 27

[73] D. Grune and C.J.H. Jacobs. Parsing techniques-a practical guide. 1990. 48

[74] A. Hall, C. Pomm, and P. Widmayer. A combinatorial approach to multi-domain sketch recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 7–14, New York, NY, USA, 2007. ACM. 35, 36

[75] A.H. Hamada. A new system for the analysis of schematic diagrams. In *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 369–372. IEEE, 1993. 22

[76] T. Hammond and R. Davis. Ladder, a sketching language for user interface developers. *Computers & Graphics*, 29(4) :518–532, 2005. 46

[77] Tracy Hammond and Randy Davis. Tahuti : A geometrical sketch recognition system for uml class diagrams, 2002. 26

[78] Tracy Hammond and Brandon Paulson. Recognizing sketched multistroke primitives. *ACM Trans. Interact. Intell. Syst.*, 1(1) :4 :1–4 :34, October 2011. 34, 36, 39

[79] X. Hilaire. *Segmentation robuste de courbes discrètes 2D et applications à la rétroconversion de documents techniques*. les-Nancy, 2004. 14

[80] Xavier Hilaire and Karl Tombre. Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 :890–904, 2006. 73

[81] R. Horaud, O. Monga, et al. Vision par ordinateur : outils fondamentaux. 1995. 30

[82] H. Hse and A.R. Newton. Sketched symbol recognition using zernike moments. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 367–370. IEEE, 2004. 38

[83] M.K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2) :179–187, 1962. 38

[84] G. Huang, W. Zhang, and L. Wenyin. A discriminative representation for symbolic image similarity evaluation. *Graphics Recognition. Recent Advances and New Opportunities*, pages 71–79, 2008. 34

[85] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification : A technique for rapid geometric design. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 105–114. ACM, 1997. 52

[86] David W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(1) :23–37, January 1996. 38

[87] J.A.P. Jorge and E.P. Glinert. Online parsing of visual languages using adjacency grammars. In *Visual Languages, Proceedings., 11th IEEE International Symposium on*, pages 250–257. IEEE, 1995. 42, 44, 49

[88] JN Kapur, P.K. Sahoo, and AKC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3) :273–285, 1985. 30

[89] Levent Burak Kara and Thomas F. Stahovich. Sim-u-sketch : A sketch-based interface for simulink. In *Proceedings of AVI-2004*, pages 354–357, 2004. 36

[90] Levent Burak Kara and Thomas F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. 27, 36

[91] T. Kasami and K. Torii. A syntax-analysis procedure for unambiguous context-free grammars. *Journal of the ACM (JACM)*, 16(3) :423–431, 1969. 47

[92] H. Kato and S. Inokuchi. A recognition system for printed piano music using musical knowledge and constraints. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 435–55. Springer-Verlag, Berlin, 1992. 21, 20

[93] Soo-Hyung Kim and Jin Hyung Kim. Automatic input of logic diagrams by recognizing loop-symbols and rectilinear connections. *IJPRAI*, 8(5) :1113–1129, 1994. 22

[94] D.E. Knuth. On the translation of languages from left to right. *Information and control*, 8(6) :607–639, 1965. 47

[95] A. Kosmala and G. Rigoll. Recognition of on-line handwritten formulas. In *6th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 1998. 38

[96] A. Kosmala, G. Rigoll, and A. Brakensiek. Online handwritten formula recognition with integrated correction recognition and execution. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 590–593. IEEE, 2000. 22

[97] H.P. Kriegel and S. Schönauer. Similarity search in structured data. *Data Warehousing and Knowledge Discovery*, pages 309–319, 2003. 38

[98] G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems : celebrating interdependence*, pages 258–264. ACM, 1994. 53

[99] James A. Landay and Brad A. Myers. Sketching interfaces : Toward more human interface design. *Computer*, 34(3) :56–64, March 2001. 26

[100] E. Lank, J. Thorley, S. Chen, and D. Blostein. On-line recognition of uml diagrams. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 356–360. IEEE, 2001. 23, 34, 37, 22

[101] S.W. Lee. Recognizing hand-drawn electrical circuit symbols with attributed graph matching. *Structured document image analysis*, pages 340–358, 1992. 40

[102] Aurélie Lemaitre and Jean Camillerapp. Text line extraction in handwritten document with kalman filter applied on low resolution image. *Document Image Analysis for Libraries, International Workshop on*, 0 :38–45, 2006. 73

[103] M. Lemaıtre, E. Grosicki, E. Geoffrois, and F. Prêteux. Layout analysis of handwritten letters based on textural and spatial information and a 2d markovian approach. 38

[104] I. Leplumey, J. Camillerapp, and C. Queguiner. Kalman filter contributions towards document segmentation. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 765–769. IEEE, 1995. 22

[105] J. Lladós, G. Sánchez, and E. Martí. A string based method to recognize symbols and structural textures in architectural plans. *Graphics Recognition Algorithms and Systems*, pages 91–103, 1998. 22

[106] Z. Lu. Detection of text regions from digital engineering drawings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(4) :431–439, 1998. 29

[107] S. Mace and E. Anquetil. A generic method for eager interpretation of on-line handwritten structured documents. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 1106–1109. IEEE, 2006. 42, 44, 49

[108] Sébastien Macé. *Composition manuscrite interactive et interprétation à la volée de documents structurés en ligne.* PhD thesis, Rennes, INSA, 2008. 4, 20, 26, 54, 57, 58, 65, 77

[109] Karl Macmillan, Michael Droettboom, and Ichiro Fujinaga. Gamera : A structured document recognition application development environment. In *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval*, pages 15–16, 2001. 23

[110] J. Mankoff, G.D. Abowd, and S.E. Hudson. Oops : A toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. *Computers & Graphics*, 24(6) :819–834, 2000. 52

[111] K. Marriott. Constraint multiset grammars. In *Visual Languages, 1994. Proceedings., IEEE Symposium on*, pages 118–125. IEEE, 1994. 42, 44, 49

[112] P. Martin and C. Bellissant. Low-level analysis of music drawing images. In *First International Conference Document Analysis and Recognition*, pages 417–425, 1991. 21, 20

[113] Matt Marx and Chris Schmandt. Putting people first : Specifying proper names in speech interfaces. In *In Proceedings of the ACM Symposium on User Interface Software and Technology, ACM*, pages 29–37, 1994. 51

[114] J. Mas, J. Llados, G. Sanchez, and J.A.P. Jorge. A syntactic approach based on distortion-tolerant adjacency grammars and a spatial-directed parser to interpret sketched diagrams. *Pattern Recognition*, 43(12) :4148 – 4164, 2010. 44

[115] C.A.B. Mello and R.D. Lins. Image segmentation of historical documents. *Visual2000, Mexico City, Mexico*, 2000. 30

[116] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. *Graphics Recognition Methods and Applications*, pages 123–134, 1996. 21, 22, 34, 40

[117] Hidetoshi Miyao and Minoru Maruyama. An online handwritten music symbol recognition system. *Int. J. Doc. Anal. Recognit.*, 9(1) :49–58, February 2007. 26

[118] F. Montreuil, E. Grosicki, L. Heutte, and S. Nicolas. Unconstrained handwritten document layout extraction using 2d conditional random fields. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 853–857. IEEE, 2009. 38

[119] H. Mouchère. Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs : Application à la reconnaissance de l'écriture manuscrite en-ligne. 2007. 66

[120] Nibal Nayef and Thomas M. Breuel. Statistical grouping for segmenting symbols parts from line drawings, with application to symbol spotting. *Document Analysis and Recognition, International Conference on*, 0 :364–368, 2011. 38

[121] Elizabeth Ng, Tim Bell, and Andy Cockburn. Improvements to a pen-based musical input system. In *Proceedings of the Australasian Conference on Computer Human Interaction*, OZCHI '98, pages 178–, Washington, DC, USA, 1998. IEEE Computer Society. 26

[122] S. Nicolas, T. Paquet, L. Heutte, et al. 2d markovian models for document structure analysis. In *11th International Conference on Frontiers in Handwriting Recognition (ICFHR08)*, pages 658–663, 2008. 38

[123] M. Notowidigdo and R.C. Miller. Off-line sketch interpretation. In *AAAI Fall Symposium*, pages 120–126, 2004. 22, 30, 52

[124] Nobuyuki Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1) :62–66, 1979. 30

[125] B. Pasternak. Processing imprecise and structural distorted line drawings by an adaptable drawing interpretation kernel. In *Proc. of DAS-94 : International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 349–365, Kaiserslautern, 1994. 22, 21

[126] Rachel Patel, Beryl Plimmer, John Grundy, and Ross Ihaka. Ink features for diagram recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 131–138, New York, NY, USA, 2007. ACM. 29

[127] Mathieu Pecot, Sebastien Macé, and Eric Anquetil. Interprétation interactive de plans d'architecture composés à main levée. In *Actes du XIème Colloque International Francophone sur l'Ecrit et le Document (CIFED'10)*, pages 185–200, 2010. 5

[128] B. Plimmer and M. Apperley. Software to sketch interface designs. In *Proceedings of the Ninth International Conference on Human-Computer Interaction (INTERACTŠ03)*, pages 73–80, 2003. 23

[129] A. Poon, K. Weber, and T. Cass. Scribbler : a tool for searching digital ink. In *Conference companion on Human factors in computing systems*, pages 252–253. ACM, 1995. 51

[130] Thierry Pun. A new method for grey-level picture thresholding using the entropy of the histogram. *Signal Processing*, 2(3) :223 – 237, 1980. 30

[131] B. G. Schunck R. Jain, R. Kasturi. *Machine Vision*. McGraw-Hill 1995, 1995. 30

[132] Jean-Yves Ramel, Guillaume Boissier, and Hubert Emptoz. A structural representation adapted to handwritten symbol recognition. In Atul Chhabra and Dov Dori, editors, *Graphics Recognition Recent Advances*, volume 1941 of *Lecture Notes in Computer Science*, pages 228–237. Springer Berlin / Heidelberg, 2000. 29

[133] R. Randriamahefa, JP Cocquerez, C. Fluhr, F. Pépin, and S. Philipp. Printed music recognition. In *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 898–901. IEEE, 1993. 21, 20

[134] J. Rekers, A. Schürr, and A. Sch Urry. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages and Computing*, 8 :27–55, 1997. 49

[135] Azriel Rosenfeld and Emily Johnston. Angle detection on digital curves. *IEEE Trans. Comput.*, 22(9) :875–878, September 1973. 37

[136] J.P. Salmon, L. Wendling, and S. Tabbone. Automatical definition of measures from the combination of shape descriptors. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 986–990. IEEE, 2005. 37

[137] PV Sankar and CU Sharma. A parallel procedure for the detection of dominant points on a digital curve. *Computer Graphics and Image Processing*, 7(3) :403–412, 1978. 37

[138] K.M. Sayre. Machine recognition of handwritten words : A project report. *Pattern Recognition*, 5(3) :213–228, 1973. 34

[139] Tevfik Metin Sezgin and Randall Davis. Hmm-based efficient sketch recognition. In *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 281–283, New York, NY, USA, 2005. ACM. 35, 37

[140] TM Sezgin and R. Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5) :500–510, 2008. 39

[141] A.C. Shaw. Parsing of graph-representable pictures. *Journal of the ACM (JACM)*, 17(3) :453–481, 1970. 48

[142] Alan C. Shaw. Parsing of graph-representable pictures. *J. ACM*, 17(3) :453–481, July 1970. 42

[143] M. Shilman, H. Pasula, and S. Russel R. Newton. Statistical visual language models for ink parsing. In *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pages 126–32, 2002. 27

[144] M. Shilman and P. Viola. Spatial recognition and grouping of text and graphics. In *Sketch-Based Interfaces and Modeling*, pages 91–95. The Eurographics Association, 2004. 35, 36

[145] M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 569–574. IEEE, 2004. 35, 36

[146] Y. Shima, T. Murakami, M. Koga, H. Yashiro, and H. Fujisawa. A high-speed algorithm for propagation-type labeling based on block sorting of runs in binary images. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 655–658. IEEE, 1990. 33

[147] A. Shio and Y. Aoki. Sketch plan : A prototype system for interpreting hand-sketched floor plans. *Systems and Computers in Japan*, 31(6) :10–18, 2000. 21, 34

[148] P. Sukaviriya, J.D. Foley, and T. Griffith. A second generation user interface design environment : The model and the runtime architecture. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 375–382. ACM, 1993. 53

[149] M. Szummer and Y. Qi. Contextual recognition of hand-drawn diagrams with conditional random fields. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 32–37. IEEE, 2004. 38

[150] S. Tabbone and L. Wendling. Recognition of symbols in grey level line-drawings from an adaptation of the radon transform. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 570–573. IEEE, 2004. 38

[151] S. Tabbone, L. Wendling, and K. Tombre. Indexing of technical line drawings based on f-signatures. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 1220–1224. IEEE, 2001. 37

[152] S. Tabbone, L. Wendling, and K. Tombre. Matching of graphical symbols in line-drawing images using angular signature information. *International Journal on Document Analysis and Recognition*, 6(2) :115–125, 2003. 38

[153] T. Takagi and M. Sugeno. Fuzzy identification of system and its applications to modelling and control. *IEEE Trans. Syst., Man, and Cyber*, 1 :5, 1985. 66

[154] Tomohiro Takagi and Michio Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1) :116–132, February 1985. 79

[155] Ernesto Tapia and Raul Rojas. Recognition of on-line handwritten mathematical expressions in the e-chalk system - an extension. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR '05, pages 1206–1210, Washington, DC, USA, 2005. IEEE Computer Society. 22

[156] M.R. Teague. Image analysis via the general theory of moments*. *JOSA*, 70(8) :920–930, 1980. 38

[157] W. Thimbleby. A novel pen-based calculator and its evaluation. In *Proceedings of the third Nordic conference on Human-computer interaction*, pages 445–448. ACM, 2004. 26

[158] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone. Stable and robust vectorization : How to make the right choices. *Graphics Recognition Recent Advances*, pages 3–18, 2000. 33

[159] Karl Tombre, Salvatore Tabbone, Loïc Pélissier, Bart Lamiroy, and Philippe Dosch. Text/graphics separation revisited. In *Proceedings of the 5th International Workshop on Document Analysis Systems V*, DAS '02, pages 200–211, London, UK, UK, 2002. Springer-Verlag. 29

[160] K. Toyozumi, T. Suzuki, K. Mori, and Y. Suenaga. A system for real-time recognition of handwritten mathematical formulas. In *Document Analysis and Recog-*

nition, 2001. Proceedings. Sixth International Conference on, pages 1059–1063. IEEE, 2001. 26

[161] S.H. Unger. A global parser for context-free phrase structure grammars. *Communications of the ACM*, 11(4) :240–247, 1968. 48

[162] Ernest Valveny and Enric Martí. Deformable template matching within a bayesian framework for hand-written graphic symbol recognition. In *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances*, GREC '99, pages 193–208, London, UK, UK, 2000. Springer-Verlag. 21

[163] P. Vaxivire and K. Tombre. Subsampling : A structural approach to technical document vectorization. In *Structure and Pattern Recognition (Post-proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition, Nahariya.* Citeseer, 1995. 33

[164] F.M. Wahl, K.Y. Wong, and R.G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, 20(4) :375–390, 1982. 29

[165] LT Watson, K. Arvind, RW Ehrich, and RM Haralick. Extraction of lines and regions from grey tone line drawing images. *Pattern Recognition*, 17(5) :493–507, 1984. 37

[166] L. Wenyin. Example-driven graphics recognition. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 821–842, 2002. 51

[167] L. Wenyin and D. Dori. A generic integrated line detection algorithm and its object–process specification. *Computer Vision and Image Understanding*, 70(3) :420–437, 1998. 52

[168] L. Wenyin and D. Dori. Genericity in graphics recognition algorithms. *Graphics Recognition Algorithms and Systems*, pages 9–20, 1998. 52

[169] K. Wittenburg and L. Weitzman. Relational grammars : Theory and practice in a visual language interface for process modeling. *Visual language theory*, pages 193–217, 1998. 48

[170] Bo Yu and Shijie Cai. A domain-independent system for sketch recognition. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 141–146, New York, NY, USA, 2003. ACM. 34

[171] Y. Yu, A. Samal, and S. Seth. Isolating symbols from connection lines in a class of engineering drawings. *Pattern recognition*, 27(3) :391–404, 1994. 40

[172] R. Zanibbi, D. Blostein, and J.R. Cordy. Baseline structure analysis of handwritten mathematics notation. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 768–773. IEEE, 2001. 22

[173] D. Zhang and G. Lu. Study and evaluation of different fourier methods for image retrieval. *Image and Vision Computing*, 23(1) :33–49, 2005. 38

[174] W. Zhang and W. Liu. A new vectorial signature for quick symbol indexing, filtering and recognition. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 01*, ICDAR '07, pages 536–540, Washington, DC, USA, 2007. IEEE Computer Society. 34

[175] W.T. Zheng and Z.X. Sun. Knowledge-based hierarchical sketch understanding. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 5, pages 2838–2843. IEEE, 2005. 23, 36, 22

Liste des figures

Liste des tableaux