Extraction et reconnaissance de primitives dans les façades de Paris à l'aide de similarités de graphes

Jean-Emmanuel Haugeard

THÈSE

Pour obtenir le grade de :

Docteur en Science de l'Université de Cergy-Pontoise
Spécialité : Sciences et Technologies de l'Information et de la Communication

Par

Jean-Emmanuel HAUGEARD

Équipe d'accueil :
Équipes Traitement de l'Information et Systèmes
CNRS UMR 8051, ENSEA, Université Cergy-Pontoise

Titre de la thèse :

Extraction et reconnaissance de primitives dans les façades de Paris à l'aide de similarités de graphes

Soutenue le 17 décembre 2010 devant le jury composé de :

Mme.	Sylvie	PHILIPP-FOLIGUET	Directrice de thèse
M.	Frédéric	PRECIOSO	Co-directeur de thèse
M.	Philippe-Henri	GOSSELIN	Co-directeur de thèse
Mme.	Christine	FERNANDEZ-MALOIGNE	Présidente du jury
M.	Patrick	LAMBERT	Rapporteur
M.	Luc	BRUN	Rapporteur
M.	Matthieu	CORD	Examinateur

Résumé

Cette dernière décennie, la modélisation des villes 3D est devenue l'un des enjeux de la recherche multimédia et un axe important en reconnaissance d'objets. Dans cette thèse nous nous sommes intéressés à localiser différentes primitives, plus particulièrement les fenêtres, dans les façades de Paris. Dans un premier temps, nous présentons une analyse des façades et des différentes propriétés des fenêtres. Nous en déduisons et proposons ensuite un algorithme capable d'extraire automatiquement des hypothèses de fenêtres. Dans une deuxième partie, nous abordons l'extraction et la reconnaissance des primitives à l'aide d'appariement de graphes de contours. En effet une image de contours est lisible par l'oeil humain qui effectue un groupement perceptuel et distingue les entités présentes dans la scène. C'est ce mécanisme que nous avons cherché à reproduire. L'image est représentée sous la forme d'un graphe d'adjacence de segments de contours, valué par des informations d'orientation et de proximité des segments de contours. Pour la mise en correspondance inexacte des graphes, nous proposons plusieurs variantes d'une nouvelle similarité basée sur des ensembles de chemins tracés sur les graphes, capables d'effectuer les groupements des contours et robustes aux changements d'échelle. La similarité entre chemins prend en compte la similarité des ensembles de segments de contours et la similarité des régions définies par ces chemins. La sélection des images d'une base contenant un objet particulier s'effectue à l'aide d'un classifieur SVM ou kppv. La localisation des objets dans l'image utilise un système de vote à partir des chemins sélectionnés par l'algorithme d'appariement.

Abstract

This last decade, modeling of 3D city became one of the challenges of multimedia search and an important focus in object recognition. In this thesis we are interested to locate various primitive, especially the windows, in the facades of Paris. At first, we present an analysis of the facades and windows properties. Then we propose an algorithm able to extract automatically window candidates. In a second part, we discuss about extraction and recognition primitives using graph matching of contours. Indeed an image of contours is readable by the human eye, which uses perceptual grouping and makes distinction between entities present in the scene. It is this mechanism that we have tried to replicate. The image is represented as a graph of adjacency of segments of contours, valued by information orientation and proximity to edge segments. For the inexact matching of graphs, we propose several variants of a new similarity based on sets of paths, able to group several contours and robust to scale changes. The similarity between paths takes into account the similarity of sets of segments of contours and the similarity of the regions defined by these paths. The selection of images from a database containing a particular object is done using a KNN or SVM classifier.

Remerciements

Je tiens à remercier en tout premier lieu Sylvie Philipp-Foliguet, Frédéric Precioso et Philippe-Henri Gosselin qui ont co-dirigé cette thèse. Durant ces 3 années, ils ont su orienter mes recherches sur de bonnes voies. Malgré des emplois du temps très chargés, ils ont trouvé du temps à me consacrer pour échanger des idées et redonner des petits coups de "boost".

Je remercie les rapporteurs de cette thèse Luc Brun et Patrick Lambert pour la rapidité avec laquelle ils ont lu mon manuscrit et l'intérêt qu'ils ont porté à mon travail. Merci également aux autres membres du jury qui ont accepté de juger ce travail : Christine Fernandez-Maloigne et Matthieu Cord. Je tiens aussi à remercier tous les membres du projet iTowns.

J'ai eu plaisir à travailler au sein du laboratoire ETIS et plus particulièrement avec l'équipe MIDI. Une pensée nostalgique pour tous les doctorants avec qui j'ai partagé de bons moments (blagues, chocolat, café, resto universitaire, billard, poker, barbecue, football, physique quantique, blague informatique, .bash_perso ...) : David G., Shuji, Alexis, Justine, Thomas, Sonia, Ludovic, Jean-Christophe, Jonathan, Guy, Auguste, Sylvain, Emmanuel, David P., Ayman, Julien, Lucile, Babar, Erbao et tous les petits nouveaux, même ceux qui ne rigolent pas à mes blagues.

Je remercie l'équipe pédagogique de l'ENSEA, Michel Leclerc et Frédéric Precioso pour leur confiance et leur aide.

Je n'oublie pas ma famille (parents, frère et soeur...) et mes amis non-doctorants qui aiment poser la question "alors tu trouves ?" ou "ça avance ta thèse ?", "à quoi sert ta thèse ?". Je tiens plus particulièrement à remercier Christophe et Audrey qui m'ont hébergé et soutenu lors de la rédaction du manuscrit ainsi que Régine, Etienne, Elise et Clément qui m'ont aussi accueilli pendant ces nombreuses années.

Enfin, je te remercie toi lecteur qui cherche ton nom ou qui t'intéresse vraiment à mon travail (noyaux sur graphes de contours, détection des fenêtres). Sache que si tu as des questions sur le contenu de ma thèse, ma "fenêtre" est grande ouverte et je te répondrai avec plaisir.

Table des matières

Table des figures

Publications de l'auteur

CONFÉRENCES INTERNATIONALES

1. J.-E. Haugeard, S. Philipp-Foliguet, F. Precioso, J. Lebrun, *Extraction of Windows in facade using Kernel on Graph of Contours*, Proceedings of 16th Scandinavian Conference on Image Analysis (SCIA 09), Lecture Notes in Computer Science, Oslo, Norvège, juin 2009.
2. J.-E. Haugeard, S. Philipp-Foliguet, F. Precioso, *Windows and Facades Retrieval using Similarity on Graph of Contours*, Proceedings of IEEE International Conference on Image Processing (ICIP 09), Le Caire, Egypte, novembre 2009.
3. J.-E. Haugeard, S. Philipp-Foliguet, P.-H. Gosselin, *Kernel on Graphs based on Dictionary of Paths for Image Retrieval*, Proceedings of IEEE 20th International Conference on Pattern Recognition (ICPR 10), Istanbul, Turquie, août 2010.

CONFÉRENCES NATIONALES

4. J.-E. Haugeard, S. Philipp-Foliguet, *Recherche d'objets par appariement de graphes combinant contours et régions*, RFIA : Reconnaissance des Formes et Intelligence Artificielle (RFIA 10), Caen, France, janvier 2010.

JOURNAL

5. J.-E. Haugeard, S. Philipp-Foliguet, P.-H. Gosselin, *Kernel on Graphs of Contours for Image Retrieval*, PRL : Pattern Recognition Letters, Soumis octobre 2010.

Chapitre 1

Contexte et motivation

1.1 iTowns

Ce doctorat s'inscrit dans le cadre du projet "iTowns" (Image-based Town On-line Web Navigation and Search Engine) de l'Agence Nationale de la Recherche. L'objectif de ce projet est de mettre au point une nouvelle génération d'outils multimédia sur le web qui mélange un navigateur 3D géographique (comme Geoportail, Google Earth, Microsoft live Earth) avec un moteur de recherche basé sur une indexation des données images/visuelles par le contenu. Ce projet gère des données images panoramiques très haute résolution acquises, avec une très grande densité spatiale, au niveau de la rue par un véhicule instrumentalisé (Fig.(1.1)).

FIGURE 1.1 – Une panoramique du flux d'images.

En collaboration avec différents partenaires (Institut Géographique National (IGN), Laboratoire LIP6, Laboratoire ETIS, Laboratoire Central des Ponts et Chaussées, Centre de Morphologie Mathématique de l'Ecole des Mines, Laboratoire Régional des Pont et Chaussées), nous souhaitons mettre en place un service en ligne de navigation immersive à la manière de ce que propose Google

avec Street View. Dans un premier temps, les utilisateurs pourront explorer le 12e arrondissement de Paris. Mais l'ambition du projet est bien d'englober rapidement la capitale avant d'étendre sa couverture à toute la France dans les prochaines années.

(a) (b)

FIGURE 1.2 – Le véhicule Stereopolis et ses caméras.

iTowns (Fig.(1.3)) est l'une des nombreuses applications concrètes qui fait suite au projet Stereopolis. Le projet Stereopolis est un programme de numérisation du territoire à l'échelle de la rue. A la manière des Google cars, l'IGN dispose d'un véhicule sillonnant les rues pour les photographier. Ce camion (Fig.(1.2)) est équipé de 12 caméras Full HD, de 3 lasers répartis sur un mât et d'un système de géoréférencement élaboré permettant une acquisition très précise. L'ambition de Stereopolis est de constituer une base de données qui pourra servir de source à diverses applications grand public et professionnelles, et dont la première partie sera le projet iTowns.

Le projet iTowns a donc pour objectif d'exploiter et d'indexer cette immense base de données des images numérisées par l'IGN. Grâce à la qualité des différents clichés (une image panoramique affiche une résolution de 20 millions de pixels) et à l'utilisation des relevés laser, la précision est de l'ordre du centimètre. Le niveau de détail est impressionnant. On peut zoomer au plus près d'une façade et découvrir les détails d'un bâtiment, lire les affiches présentes sur les vitrines. Dans iTowns, la navigation est enrichie avec la base de données GeoNames (base mondiale qui

FIGURE 1.3 – iTowns (Image-based Town On-line Web Navigation and Search Engine) projet de l'Agence Nationale de la Recherche. www.itowns.fr

répertorie plus de 8 millions de points d'intérêt), que l'on peut activer d'un clic. Les noms des lieux et monuments alentour s'affichent en surimpression de l'image, et il suffit de cliquer dessus pour s'y rendre directement. Outre la navigation, iTowns intègre des outils qui permettent, à l'aide de la souris, de calculer très facilement la surface d'une façade ou la largeur et la hauteur d'une fenêtre. Autre point fort du système, la possibilité d'extraire chaque élément d'une image, comme le marquage au sol, le mobilier urbain, les piétons, etc. Ainsi, grâce au moteur de recherche intégré, on peut par exemple taper le nom d'un magasin ou donner une image de la boutique et être projeté immédiatement devant son entrée.

Le projet souhaite aller plus loin en proposant divers services multimédia : on pourra par exemple, souhaitant retrouver un restaurant chinois dont on a oublié l'adresse exacte mais dont on se souvient qu'il a une façade rouge et se trouvant dans un quartier précis, taper "restaurant chinois rouge" avec le nom du quartier et ainsi retrouver le lieu. iTowns possède aussi un éditeur multimédia grâce auquel il est possible d'ajouter du contenu interactif (vidéos, liens hypertexte...) en quelques clics. Par exemple, un bandeau animé est placé sur la façade de l'opéra Bastille et présente le programme des spectacles. Pour le moment, cet éditeur sera réservé aux établissements publics, mais il pourrait être ouvert aux professionnels du privé (notamment les commerçants), voire au grand public une fois un système de modération mis en place. Après le 12e arrondissement, iTowns devrait rapidement proposer l'ensemble de la capitale, avant, d'ici quelques années, de couvrir la France.

Nous pouvons dégager et découper ce projet en deux objectifs scientifiques :

1. Le premier objectif est de naviguer de manière fluide, libre et immersive, dans un flux d'images panoramiques (sans modèles 3D) dans de très grandes collections de données de manière à voir et visiter la ville comme si nous y étions.

2. Le deuxième objectif est de construire à partir des images un système d'information basé sur le contenu (image) de manière à proposer au sein même du navigateur des services simples ou complexes basés sur des requêtes (aller à une adresse donnée, générer une carte de navigation enrichie avec de l'image, trouver la localisation d'une image apportée par l'utilisateur, sélectionner les images contenant tel objet, etc.).

   Pour atteindre ces objectifs, le projet iTowns doit relever trois défis :

   – Le premier défi est de visualiser et de naviguer à travers le web à l'intérieur de très grands volumes d'images panoramiques géoréférencées, acquises par le système de cartographie mobile de l'IGN (Fig.(1.2)). Dans le cadre de ce projet, un Teraoctet de données acquises sur la ville de Paris sera exploité. Cela correspond à 25000 vues panoramiques (composées de dix images HD chacune) le long d'une centaine de kilomètres linéaires de rues.

   – Le deuxième défi consiste à extraire des images de manière complètement automatique et en un temps raisonnable autant de primitives, objets simples et complexes que possible ainsi que les relations géométriques et topologiques entre objets pour une indexation par le contenu.

   – Le troisième défi consiste à exploiter et combiner les différents objets, primitives et autres signes précédemment extraits afin de construire des systèmes d'apprentissage efficaces sur ces données permettant des comparaisons et des classifications à un haut niveau sémantique. Des stratégies de recherche par le contenu permettront alors de fournir des services de fouille de données intelligents avec différents niveaux de complexité.

Dans cette thèse nous allons nous intéresser plus particulièrement au deuxième défi et la reconnaissance d'objets dans les façades de Paris.

1.2 La reconnaissance d'objets et l'indexation d' images

Le domaine de l'image numérique est un domaine en pleine expansion. Depuis quelques années, avec l'explosion d'Internet et aussi le développement à grande échelle de la photographie numérique, il n'est pas rare d'avoir des bases d'images numériques contenant plusieurs millions (voire milliards) d'images (exemple : Flickr avec plus de 4 milliards d'images en 2010), que ce soit des bases ciblées pour un domaine d'activité professionnelle (journalisme, tourisme, éducation, musées, ...) ou tout simplement des bases de particuliers qui accumulent

d'immenses quantités de photographies numériques (souvenirs, voyages, famille, événements, ...).

Il faut donc trouver des méthodes permettant de gérer et de rechercher de l'information dans ces bases d'images. Il s'agit d'un sujet de recherche très actif dans les communautés de recherche en apprentissage automatique et vision par ordinateur depuis plus d'une vingtaine d'années. Ce sujet hérite des connaissances et des méthodes accumulées dans d'autres domaines liés à l'image, entre autres pour la reconnaissance des formes. En reconnaissance des formes, le but est d'identifier le contenu d'une image, de catégoriser, de trouver des caractéristiques permettant d'indexer l'image et de retrouver des images similaires lors d'une requête en fonction du contenu des images. L'indexation et la recherche d'images héritent des nombreuses problématiques de la reconnaissance de forme. Cependant la recherche interactive s'oriente vers la caractérisation de grandes bases, avec peu d'exemples, et la possibilité de faire intervenir des utilisateurs lors de la recherche.

Un problème resté ouvert dans cette recherche concerne l'identification d'informations à caractère sémantique dans l'image. En effet, une image numérique est avant tout un signal représenté le plus souvent sous la forme d'une matrice à deux dimensions. Pour chaque élément de l'image, ou pixel, nous avons une information, en niveau de gris ou en couleur (rouge-vert-bleu ou autre codage), d'un élément de la scène que nous observons. De cette matrice ne contenant que des nombres, nous voulons extraire des informations sur le contenu sémantique réel de la scène (exemple figure (Fig.(1.4)). Par exemple, une telle matrice de nombres peut représenter une scène contenant un paysage, composé de montagnes, de mers ou d'édifices urbains avec des personnes en avant-plan que nous souhaitons identifier.

Il existe en fait deux fossés majeurs pour un logiciel essayant d'interpréter automatiquement le contenu d'une image :

- Le fossé sensoriel est défini comme le fossé existant entre le monde réel 3D et sa représentation en une image 2D. En effet, la réalité qui nous entoure existe en trois dimensions. La prise d'une photographie de cette réalité se fait aujourd'hui en deux dimensions seulement. Ceci résulte donc en une perte nette d'information dont il faut tenir compte lorsqu'on tente d'analyser le contenu d'une image.

- Le fossé sémantique est défini comme le fossé entre la représentation bas niveau d'une image et l'interprétation haut niveau que les humains en font. Une image est une matrice de nombres où chaque nombre, ou pixel,

représente une intensité lumineuse en un point de l'espace. Elle est définie principalement en termes mathématiques et physiques. Par contre, lorsque nous, humains, regardons ces images, nous voyons les objets, en termes sémantiques, qui se trouvent dans les images (les gens, les paysages, les objets).

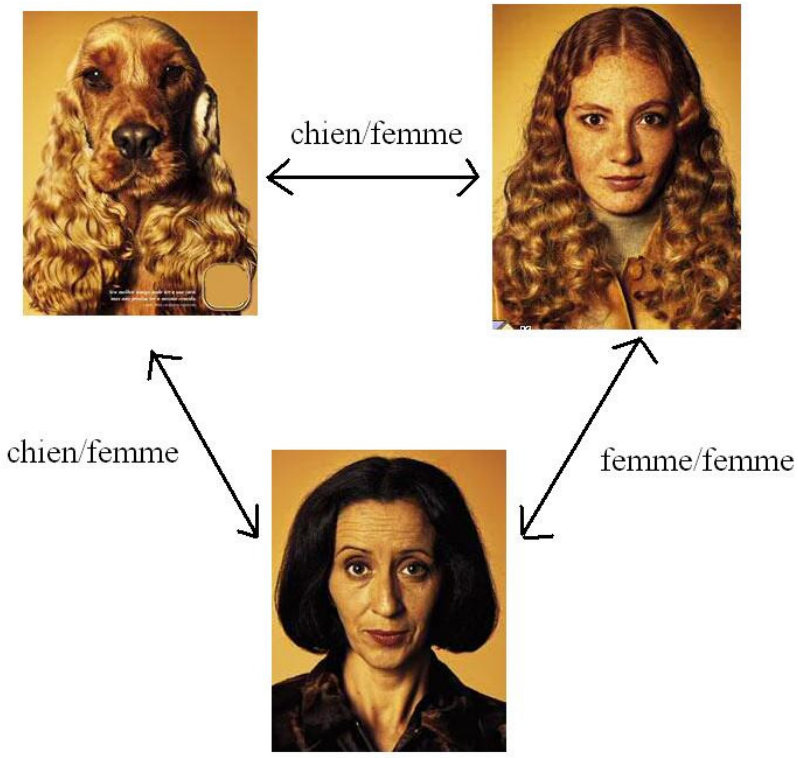


FIGURE 1.4 – Comment caractériser une image, matrice à deux dimensions, pour que le programme interprète la bonne sémantique de l'image et puisse indexer et classer correctement ces images. L'information texture et couleur sur les oreilles du chien est assez proche des informations sur les cheveux de la demoiselle.

Il existe donc un énorme décalage entre trois niveaux : la représentation que se fait l'ordinateur d'une image, l'observation de cette image par les humains et la réalité qui est représentée par cette image. Le décalage le plus grand est créé par l'ordinateur dans ses modes de représentation. Dans cette thèse, nous nous intéressons au deuxième fossé, concernant l'aspect sémantique. En effet, nous, humains, arrivons à saisir le sens d'une image 2D que nous observons, ce qui signifie que le fossé sensoriel ne perturbe pas notre compréhension des images et la réalité

qu'elles représentent. La recherche d'images par similarité (CBIR : content-based image retrieval) est un grand challenge de ces vingt dernières années. Plusieurs voies complémentaires sont possibles pour combler le fossé sémantique dans un processus d'indexation et/ou de recherche de forme.

Une voie de plus en plus utilisée pour pallier le manque de connaissance dans un système de vision par ordinateur est de concevoir des systèmes semi-automatiques (par exemple : RETIN système de recherche d'images par le contenu du laboratoire ETIS). L'interaction homme/machine permet de faire un apprentissage par renforcement. L'utilisateur fournit en entrée du système une image, dite image requête, et souhaite trouver d'autres images qui lui sont similaires. La notion de similarité reste à définir mais, en général, c'est une similarité visuelle en couleur, texture et/ou forme qui est utilisée. Eventuellement, la fonction de similarité peut être affinée en fonction des besoins de l'utilisateur avec des techniques dites de "retour de pertinence", connues en anglais sous le nom de *relevance feedback*. Nous supposons que l'humain maîtrise la sémantique et qu'avec l'aide de quelques interactions simples et ciblées (par exemple, l'identification d'exemples positifs et négatifs à partir d'un premier retour du programme), le système pourra apprendre ce que l'utilisateur veut retrouver.

Afin de réduire le fossé sémantique, nous souhaitons dans cette thèse proposer des méthodes de reconnaissance et d'extraction d'objets dans les façades de bâtiments lors de l'annotation automatique d'images.

(a) (b)

FIGURE 1.5 – **Pouvez-vous identifier les objets présents dans ces deux images de contours ?**A partir de la seule information des contours, notre système visuel est capable de percevoir, d'interpréter et de reconnaitre les différents objets. Les objets évidents à identifier sont donnés sur la figure (Fig.(2.19)) dans le chapitre 2.

1.3 Notre démarche

Considérons les images de la figure (Fig.(1.5)), et tentons d'identifier les objets présents. Les identités des objets nous paraissent évidentes. Cette simple expérience illustre le fait que les contours peuvent être utilisés pour reconnaître des objets dans les images. Des études psychophysiques telles que celles de Biederman et Ju [BJ88] confirment cette hypothèse. Le système visuel humain est capable d'interpréter une image et de reconnaître les objets présents dans l'image avec le peu d'information portée par les contours principaux. L'information contenue dans les images couleurs avec 24 bits par pixel est considérablement réduite à une image binaire de contours, mais néanmoins l'interprétation reste encore évidente. En partant de cette intuition, nous avons décidé de construire un système automatique de reconnaissance d'objets qui utilise uniquement les contours principaux. Une information importante incluse dans les images de contours est la relation spatiale entre les différents contours, ce qui est souvent qualifiée de groupement perceptuel (des contours). De plus, nous constatons sur la figure (Fig.(1.5)) que

les fenêtres sont des objets importants des façades. Afin de réduire le fossé sémantique, nous nous sommes intéressés à la caractérisation des objets présents dans les façades et notamment les fenêtres. Nous avons donc cherché à extraire des objets simples puis à les caractériser par des ensembles de contours afin de pouvoir les comparer et les classer. Afin de pouvoir comparer des ensembles de contours, nous avons décidé de structurer ces ensembles en formant des graphes de contours. Nous sommes confrontés alors à un problème d'appariement inexact de graphes. Notre objectif est de déterminer les similarités entre des images ou des formes à partir de leur représentation par graphes de contours.

L'approche proposée dans cette thèse est illustrée sur la figure (Fig.(1.6)). Nous pouvons la voir comme constituée de plusieurs parties :

- extraction d'objets simples
- appariement de graphes et apprentissage des façades
- extraction de sous-graphes dans un graphe.

Nous présentons dans le chapitre 2 l'extraction d'hypothèses de fenêtres. Comme le montre l'état de l'art (section 2.1.1), différentes techniques cherchent à extraire des fenêtres afin d'apporter de l'information sur le style de fenêtre ou la modélisation 3D des fenêtres. En nous basant sur une de ces méthodes due à Lee et Nevatia [LN04], nous proposons une méthode d'extraction automatique d'hypothèses de fenêtres. Cependant l'extraction conduit à de nombreux faux positifs. Afin de classer les fenêtres et d'écarter les faux positifs, nous souhaitons apprendre un modèle de fenêtre. Ce modèle de fenêtre est décrit par un ensemble de contours représentant les fenêtres. Avant d'introduire les techniques d'apprentissage et les différents techniques d'appariements de graphes (chapitre 3), nous décrivons dans la section 2.1.2 nos façades et hypothèses de fenêtres comme des graphes relationnels attribués de contours. Nous introduisons quelques définitions sur les graphes et les relations topologiques, puis nous décrivons les façades comme des graphes relationnels attribués de contours.

Dans le chapitre 3, nous réalisons un état de l'art sur les techniques d'appariements de graphes. Parmi toutes ces techniques, nous avons choisi de nous intéresser tout particulièrement à une similarité par fonction noyau, qui offre de nombreux avantages pour l'analyse et l'exploitation des graphes dans le cadre de l'apprentissage. Nous réalisons un état de l'art sur les techniques d'apprentissages à noyaux sur graphes. Puis nous proposons dans le chapitre 4, un cadre théorique de mise en correspondance de graphe de contours basé sur un noyau sur graphes de contours. Nous proposons une nouvelle similarité de chemins de contours à l'aide d'un noyau sur chemins. La recherche des meilleurs appariements de chemins nous permet de donner une valeur de similarité entre différents graphes

et de classer les images. D'une part, nous introduisons et utilisons les graphes globaux de contours pour classer des imagettes contenant des hypothèses de fenêtres selon leurs similarités. D'autre part, nous nous affranchissons de l'étape d'extraction d'hypothèses de fenêtres et exploitons directement le graphe des façades pour localiser la position des sous-graphes fenêtres. Nous cherchons donc à apparier les graphes requêtes avec des sous-graphes des façades. Pour celà, nous proposons de mixer l'approche contours et une approche région autour de chemins d'intérêt. Enfin, pour réduire la complexité de calcul, nous proposons une dernière méthode basée sur un dictionnaire de chemins de contours et d'un noyau incrémental. Différentes expérimentations ont été menées et sont présentées dans le chapitre 5.

Dans le chapitre 5, nous discutons de l'implémentation et de la représentation par arbre de recherche des noyaux sur graphes. Cette représentation nous permet d'utiliser l'algorithme de "séparation et évaluation" afin de réduire la complexité des calculs. Puis nous évaluons le modéle mis en place dans le chapitre 4 pour comparer des graphes de contours.

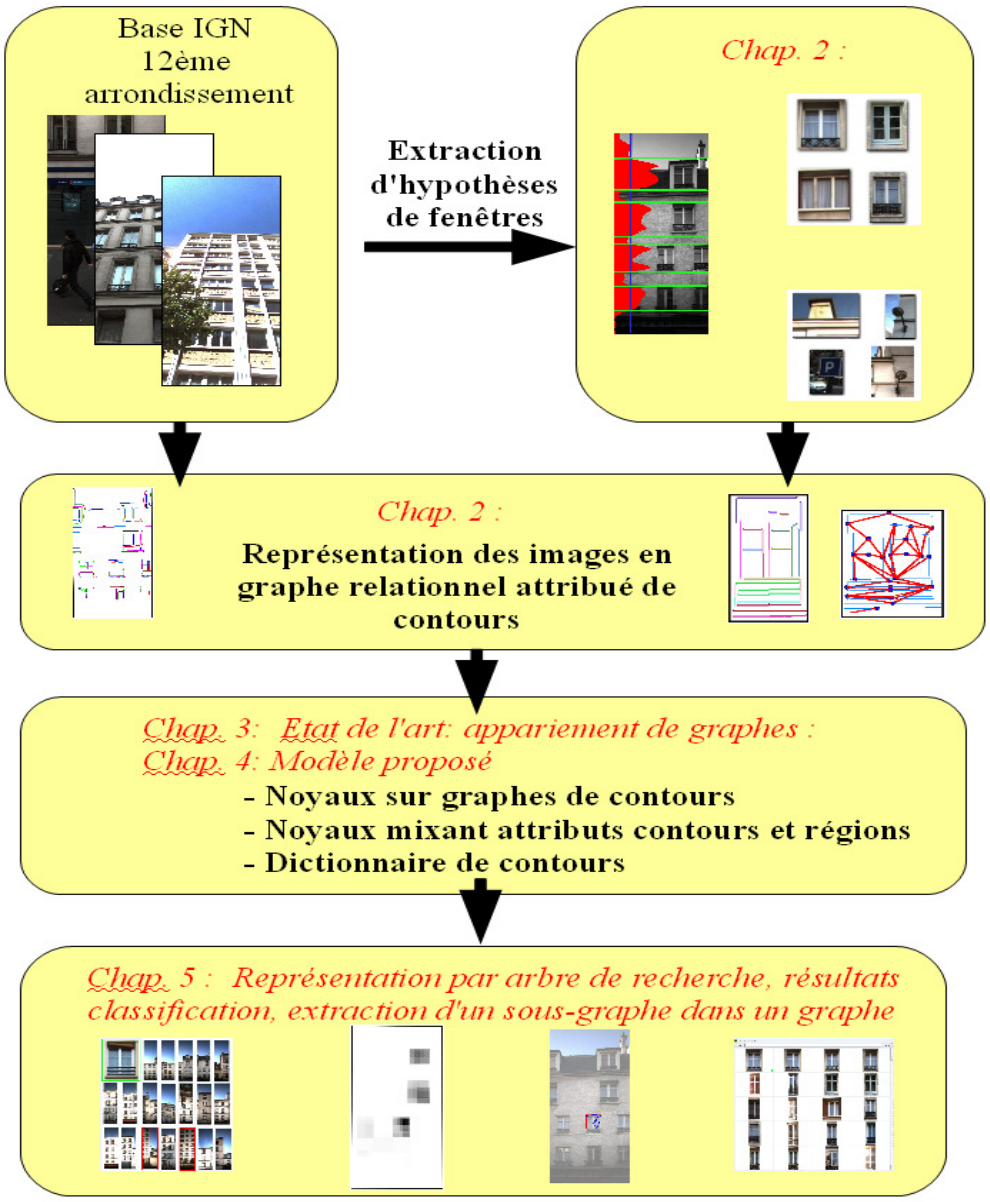


FIGURE 1.6 – Notre démarche.

Chapitre 2

Analyse et représentation des images de façades

Afin de représenter les images, la première étape que nous proposons est d'analyser les façades présentes dans les images et plus particulièrement l'objet fenêtre. La fenêtre est un élément constitutif important des façades. Après un état de l'art sur la détection d'hypothèses de fenêtres, nous avons choisi de nous concentrer principalement sur l'amélioration et l'automatisation de l'algorithme de détection de Lee et Nevatia [LN04]. De plus nous introduisons la notion d'ensemble de contours pour analyser l'information architecturale présente dans les façades. Nous travaillons sur l'idée de groupement perceptuel de contours et ainsi représenter les images par un Graphe Relationnel Attribué de contours.

2.1 Détection d'hypothèses de fenêtres

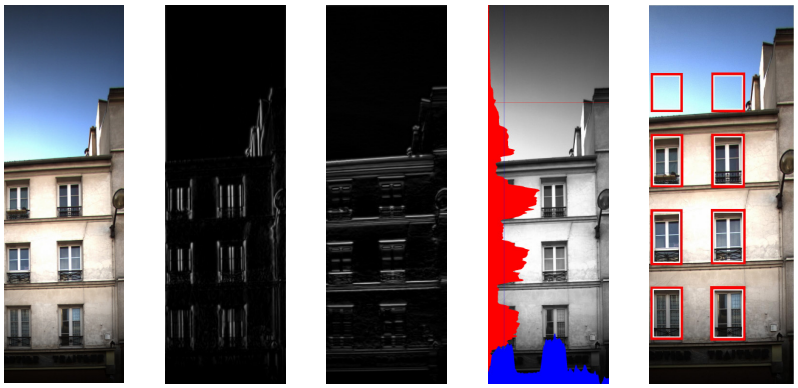
2.1.1 Etat de l'art

Ces quinze dernières années, les bâtiments étaient modélisés de façon grossière (polyhèdre...) et étaient pauvres en informations sémantiques et caractérisations texturelles. Le besoin d'améliorer la visualisation et l'extraction d'information des façades a contribué à orienter les recherches sur l'extraction de primitives dans les façades. Bien que les travaux sur l'extraction des fenêtres et autres entités architecturales soient récents et peu nombreux, ils se rejoignent tous sur le fait qu'une façade possède des caractéristiques particulières (symétrie, alignement des entités) et ils décrivent une fenêtre comme un élément de forme rectangulaire avec un ratio hauteur/largeur défini et se trouvant aligné avec les autres fenêtres. Profitant de cet alignement, la plupart des auteurs proposent d'en tirer un histogramme vertical et horizontal (accumulation des points 3D, accu-

mulation du gradient, information mutuelle) puis d'extraire les fenêtres ou autres entités architecturales (étages, portes) à partir de ces histogrammes.

2.1.1.1 Accumulation et histogrammes des gradients verticaux et horizontaux

Dans [LN04], Lee et Nevatia utilisent l'alignement des fenêtres dans les façades et leurs propriétés géométriques. Ils supposent qu'une fenêtre est rectangulaire et se trouve alignée verticalement et horizontalement aux autres fenêtres de la façade. Partant de ce constat, ils proposent une méthode basée sur l'accumulation et la projection des normes en x et en y du gradient selon l'axe horizontal et vertical (Fig.(2.1)). Grâce à l'alignement des fenêtres, des accumulations se forment alors sur les histogrammes horizontal et vertical. Les hypothèses des fenêtres peuvent être extraites ensuite par intersections de ces accumulations sur le profil vertical et le profil horizontal. Les résultats qu'ils obtiennent sont bons et précis sur des bases simples où les fenêtres sont régulièrement alignées et les murs uniformes et sur les façades sans problèmes d'occlusion et/ou de variation de luminosité.



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

FIGURE 2.1 – **Extraction des hypothèses de fenêtres (a)** Image originale **(b)** Norme du gradient en x **(c)** Norme du gradient en y **(d)** Accumulation et histogrammes vertical et horizontal **(e)** Résultat. Ces résultats sont obtenus à l'aide de l'algorithme de [LN04] sur une image redressée de la base iTOWNS.

2.1.1.2 Découpage de manière hiérarchique

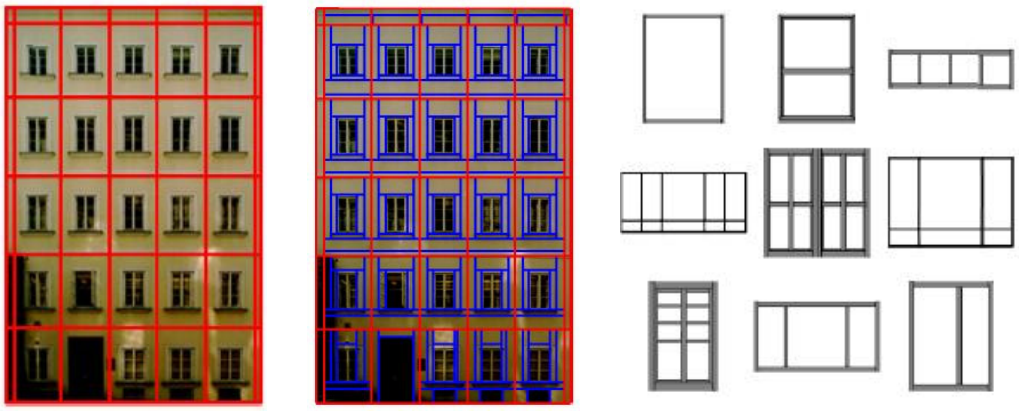
Une autre approche consiste à représenter la façade par une structure hiérarchique des éléments qui composent la façade. Müller *et al.* [MZWG07] , Brenner *et al.* [BR06], Alegre *et al.* [AD04] et Moslah *et al.* [MVNT+10] utilisent des

modèles procéduraux afin de découper de manière hiérarchique "top/down" les façades en sous-éléments tels que les étages, fenêtres et portes. A partir d'une image de façade rectifiée, les auteurs la transforment en un modèle contenant un arbre sur la structure sémantique de la façade. Cette transformation se déroule en 3 étapes :

- Détection de la structure de la façade (Fig.(2.2 a)) : Un algorithme basé sur l'information mutuelle permet de détecter les diverses symétries contenues dans la façade. La façade est alors divisée en étages et chaque étage en tuiles. Une tuile est un découpage régulier de la façade. Elle représente un élément architectural comme par exemple une fenêtre.

- Précision des tuiles (Fig.(2.2 b)) : Les tuiles sont regroupées par similarité à l'aide d'un algorithme de clustering, puis segmentée en régions rectangulaires. Un cluster représente un type de fenêtre ou de porte.

- Reconnaissance des éléments architecturaux (Fig.(2.2 c)) : A partir d'une base prédéfinie d'éléments architecturaux, les auteurs recherchent les meilleurs appariements entre chaque prototype des clusters et les éléments de la base. Ils détectent ainsi les éléments structurant la façade et proposent un modèle basé sur cette structure sémantique.



(a) (b) (c)

FIGURE 2.2 – **Division de la façade : Approche procédurale. (a)** Découpage de la façade en étages et tuiles à l'aide des propriétés de symétrie. **(b)** Segmentation des tuiles. **(c)** Bases d'éléments architecturaux servant pour le matching avec les tuiles segmentées. Figures issues de [MZWG07].

2.1.1.3 Détection de fenêtres basée sur un modèle implicite de forme

Mayer et Reznik [MR07], Dick *et al.* [DTC04] et Moslah *et al.* [MVNT+10] proposent de définir un modèle paramétrique de fenêtres. En changeant les paramètres de leur modèle (comme la largeur, le ratio hauteur/largeur, la luminosité, etc.) et à l'aide de méthodes de Monte-Carlo par chaine de Markov (MCMC), ils tentent de créer l'image qui est la plus similaire à l'image donnée. Par exemple, [MR07] utilise des "patchs" de points d'intérêt (points de Förstner) pour apprendre un modèle implicite d'une fenêtre qui est ensuite réutilisé pour extraire des hypothèses de fenêtres par le biais de méthodes MCMC.

Suite à de nombreux tests, Mayer [MR07] caractérise une fenêtre par :

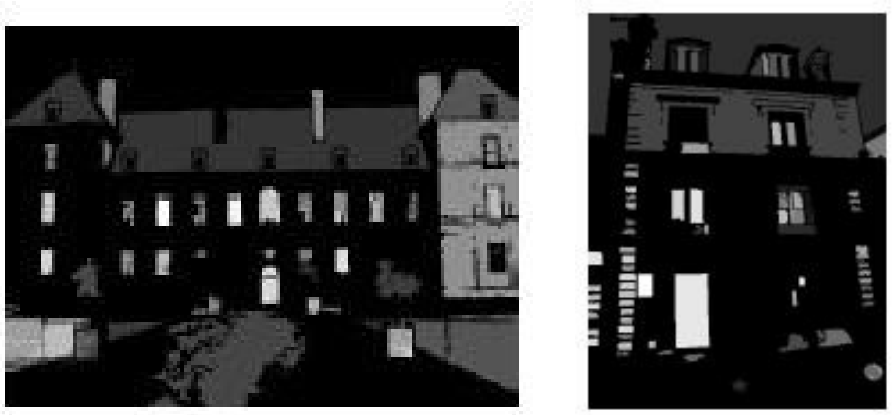
- La plupart des fenêtres sont rectangulaires.
- Le ratio hauteur/largeur d'une fenêtre est généralement compris entre 0,25 et 5.
- Les fenêtres sont fréquemment sombres, et plus particulièrement pour le canal rouge. En effet les fenêtres étant constituées de verre, le rouge la traverse, tandis que le bleu du ciel est reflété.

A l'aide de ces mêmes caractéristiques combinées à des techniques de morphologie mathématique, Hernandez et Marcotegui [HM08] facilitent l'interprétation des façades et l'extraction de grammaire en segmentant la façade et en faisant ressortir les éléments architecturaux. Les auteurs proposent d'utiliser des algorithmes d'ouverture ultime avec des contraintes de formes. Ils définissent ainsi une fonction de similarité pour la structure interne de la façade à partir des caractéristiques propres aux fenêtres (Mayer [MR07]).

Les résultats (Fig.(2.3)) obtenus donnent une bonne segmentation des fenêtres malheureusement d'autres entités, comme les briques par exemple, ressortent aussi de cette segmentation.

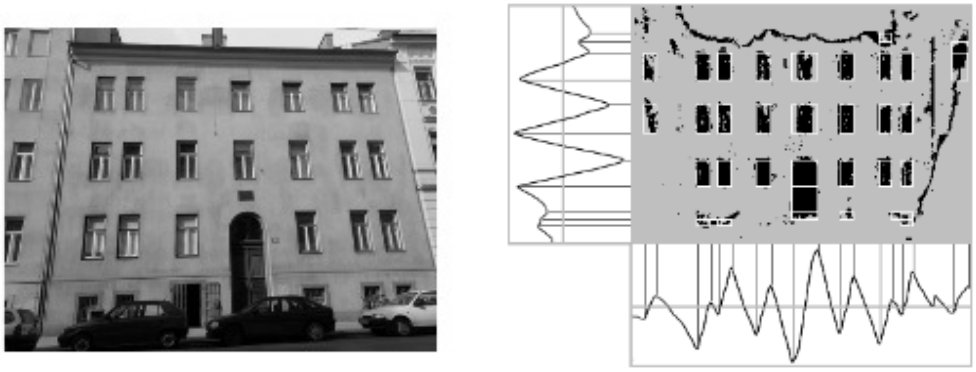
2.1.1.4 Densité des points 3D

Dans les méthodes précédentes, les auteurs utilisaient l'alignement et la forme des fenêtres pour les détecter. Dans le contexte de modélisation des bâtiments, une autre méthode [SB03] souligne le fait que les fenêtres ne se trouvent pas dans le même plan que la façade. Werner et Zisserman [WZ02] et Schindler et Bauer [SB03] détectent les fenêtres comme des objets situés en renforcement par rapport au plan de la façade. Schindler [SB03] propose un algorithme de détection de fenêtre basé sur des points 3D qui ont été extraits de l'image correspondante, ce qui nécessite au moins 2 images de la même façade.



(a) (b)

FIGURE 2.3 – **Résultats de segmentation de façades à partir d'un algorithme d'ouverture ultime.** On note une sur-segmentation de la façade, les fenêtres ressortent bien, mais des éléments ,comme les briques, sont sur détectés. Résultats issus de [HM08].



(a) (b)

FIGURE 2.4 – **Détection d'hypothèse de fenêtre.a** Une des 4 images de la séquence. **b** Plan de la façade rectifiée avec la projection des "outliers" et les hypothèses de fenêtres(rectangle). En bas et à gauche, les fonctions de la direction du gradient en x et y. Figures issues de [SB03].

Dans cette approche, ils recherchent les "outliers" (points qui ne sont pas dans le plan de la façade) qu'ils projettent ensuite sur l'image 2D. Les zones denses en points sont des hypothèses de fenêtres. Puis ils réalisent le découpage à l'aide d'un histogramme des directions du gradient en x et y (Fig.(2.4)). Ils profitent eux aussi de l'alignement des fenêtres pour calculer les histogrammes verticaux et horizontaux.

2.1.1.5 Apprentissage des fenêtres

Une nouvelle approche basée sur l'apprentissage de nombreux exemples à différentes échelles et différentes luminosités est proposée par Johansson [Joh02] et Ali *et al.* [ASJ+07]. Les auteurs proposent d'utiliser les propriétés de forme d'une fenêtre et de la décrire par un ensemble de descripteurs extraits à l'aide des ondelettes de Haar. Leurs travaux diffèrent ensuite dans la technique d'apprentissage de ces descripteurs. Johansson [Joh02] propose dans sa thèse d'utiliser les séparateurs à vastes marges (support vector machine SVM Vapnik [Vap98]), tandis que le modèle de détection de fenêtres d'Ali *et al.* [ASJ+07] est basé sur un apprentissage par boosting (Viola et Jones [VJ01] et Lienhart *et al.* [LKP03]).

Profitant de la forme rectangulaire d'une fenêtre, les auteurs (Ali *et al.* [ASJ+07]) représentent une fenêtre par un ensemble de descripteurs de Haar. Leur classifieur fort est défini en utilisant Gentle Adaboost de Freund et Schapire [FS96] et se compose d'une cascade de classifieurs faibles. Au cours de la phase d'apprentissage supervisé un sous-ensemble de caractéristiques (classifieurs faibles), qui distinguent fenêtre et non-fenêtre, est extrait. Puis dans la phase de détection, une fenêtre de taille variable (multi-échelle) est déplacée sur l'image. Les classifieurs faibles sélectionnés par Gentle Adaboost, sont alors calculés. Si le contenu de l'image à l'intérieur de la fenêtre glissante passe toutes les étapes de la classification, il est alors déclaré comme fenêtre de façade. Pour l'apprentissage et l'évaluation de leur système de détection de fenêtre, ils utilisent les bases de données référencées dans la littérature sur la reconnaissance de bâtiment (Zurich building base [ZuB], Graz base [TSGa],[TSGb]).

Les résultats obtenus par Johansson [Joh02] et Ali [ASJ+07] sont satisfaisants, répondant à leur problématique selon laquelle au moins quelques fenêtres par bâtiments doivent être détectées. Cependant il faut noter que leur système a un taux de vrais négatifs élevé et donne de bons résultats uniquement sur les vues de face.

2.1.1.6 Conclusion

Les résultats obtenus des différentes approches sont satisfaisants sur des bases simples (façade bien symétrique, fenêtre simple et rectangulaire, fenêtres alignées horizontalement et verticalement...). Dans le cadre des villes historiques anciennes comme Paris l'architecture est différente et les images sont plus complexes : les fenêtres ne sont pas nécessairement alignées (Fig.(2.5a)), les textures ne sont pas uniformes, il y a des variations de luminosité et de nombreuses occlusions dûes aux arbres, voitures, etc... De plus l'utilisation de nombreux paramètres sont utiles pour segmenter la façade. Afin d'extraire les fenêtres, nous proposons un nouvel algorithme automatisé inspiré de celui de Lee et Nevatia [LN04].

2.1.2 Notre approche

Dans l'état de l'art précédent, nous remarquons que la plupart des auteurs utilisent le fait que les fenêtres sont alignées verticalement et horizontalement dans les façades. De plus ils utilisent aussi la forme rectangulaire de la plupart des fenêtres. Dans un premier temps, nous avons donc décidé d'utiliser la méthode de Lee et Nevatia [LN04] basée sur l'accumulation et projection des gradients en x et en y. Etant donné que les façades sont découpées en étages, les fenêtres sont généralement alignées horizontalement. Nous proposons donc d'extraire d'abord les étages puis de travailler séparément sur chacun d'eux pour extraire des fenêtres ou du moins des rectangles qui sont susceptibles d'être des fenêtres. De plus, nous améliorons la méthode en automatisant l'extraction d'hypothèses de fenêtres par la recherche du meilleurs poids d'échelle. En effet, dans l'article de Lee et Nevatia [LN04], ils fixent les mêmes paramètres de lissage et de dérivation pour toutes les façades. Or des réglages adaptés à la texture de la façade pourraient fournir de bien meilleurs résultats.

Principe général

Afin de situer les étages, nous calculons à l'aide d'un filtre dérivateur les composantes horizontales du gradient(Fig.(2.5 b)), puis nous projetons horizontalement et accumulons les normes pour former un histogramme (Fig.(2.5 c)). La répartition des composantes horizontales du gradient est ainsi représentée dans cet histogramme et permet de visualiser plusieurs classes. En effet nous constatons des hautes valeurs correspondant plus ou moins aux fenêtres d'un étage tandis que les basses valeurs représentent les murs, toit, ciel... Pour séparer les classes (étages/pas étages), l'histogramme est ensuite seuillé par rapport à sa valeur moyenne. La façade est ainsi divisée en étages (Fig.(2.5 d)). Le processus est répété dans l'autre sens séparément pour chaque étage, donnant alors les hypothèses de fenêtres.

Extraction automatique des étages

Etant donné que nous avons besoin d'un ensemble précis de contours, nous utilisons les opérateurs de lissage et de dérivation de Shen-Casten [SC92]. Shen et Castan ont proposé des opérateurs optimisant un critère incluant la bonne détection et la bonne localisation. Comme le souligne Cord *et al.* [CHP97], les filtres de Shen-Castan offrent un meilleur compromis pour une bonne localisation et bonne détection que les filtres de Canny.

$$f(x) = c \times e^{-\beta|x|}$$

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

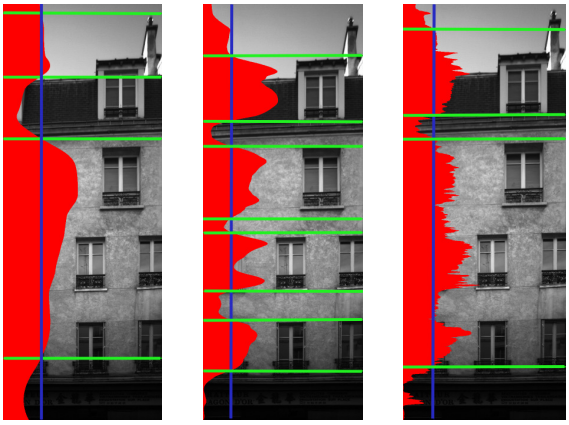
FIGURE 2.5 – **Extraction d'hypothèses de fenêtres. (a)**Exemple de façade où les fenêtres ne sont pas allignées verticalement. **(b)**Normes des composantes horizontales du gradient. **(c)**Projection horizontale et accumulation des normes. **(d)**Division en 4 étages selon la moyenne (trait bleu sur (c)). **(e)**Projection verticale et accumulation des normes des composantes verticales du gradient. **(f)**Hypothèses de fenêtres.

$$\text{Le filtre de dérivation est : } h(x) = \begin{cases} d \times e^{-\beta x} & \text{si } x \geq 0 \\ d \times e^{\beta x} & \text{si} x < 0 \end{cases}$$

Ces filtres de lissage et de dérivation dépendent du paramètre β. Ce paramètre définit la "largeur" du filtre : plus β est petit, plus le lissage effectué par le filtre est important. Le niveau de détails des contours dépend donc de ce paramètre.

Si le lissage est trop fort, certains contours disparaissent (Fig.(2.6)). Au contraire si le lissage est trop faible, il y a alors trop de bruit (texture entre les fenêtres). En fonction du paramètre β l'histogramme est plus plus ou moins précis. Le nombre d'étage sélectionné est alors choisi sur l'histogramme seuillé par rapport à la valeur moyenne des accumulations sur l'histogramme. Le nombre d'étages extraits dépend donc du paramètre β, mais n'évolue pas régulièrement avec β (Fig.(2.7)). Sur la courbe d'évolution du nombre d'étages extraits par rapport au paramètre β, nous observons un plateau qui est un bon compromis entre trop de bruit et trop de contours.

Afin de déterminer la valeur du β correspondant à ce plateau, nous calculons un score S_{β_i} pour chaque valeur de β_i (β_i évoluant entre 0 et 1). L'idée est de maximiser un score dépendant de la stabilité de l'histogramme (même nombre de pics) et l'amplitude des pics H_{pj}. Nous définissons S_{β_i} :



(a) $\beta = 0.01$ (b) $\beta = 0.07$ (c) $\beta = 0.1$

FIGURE 2.6 – **Le nombre d'étages dépend du paramètre de lissage et de dérivation β.** Ce nombre d'étage est extrait à l'aide de l'histogramme des composantes horizontales du gradient. L'histogramme est seuillé par rapport à sa valeur moyenne (trait bleu). **(a)**Fort lissage. **(b)**Bon compromis. **(c)**Faible lissage. **(d)**Evolution du nombre d'étages en fonction de β

$$S_{\beta_i} = \begin{cases} \overbrace{\frac{p_{\beta_{i-1}}}{p_{\beta_i}}}^{Stabilit} \cdot \frac{\overbrace{\sum_{j=1}^{p_{\beta_i}} maxH_{pj}}^{amplitudemoyennedespics}}{p_{\beta_i}} & \text{si } p_{\beta_{i-1}} < p_{\beta_i} \\ \frac{\sum_{j=1}^{p_{\beta_i}} maxH_{pj}}{p_{\beta_{i-1}}} & \text{sinon} \end{cases} \tag{2.1}$$

avec p_{β_i} le nombre de pics pour β_i.

Découpage des étages en hypothèses de fenêtres.

Pour chaque étage extrait, nous répétons séparément le même processus dans l'autre sens. Nous utilisons le filtre de Shen-Castan pour les composantes verticales du gradient, puis nous réalisons une projection verticale de ces normes donnant alors les hypothèses de fenêtres (Fig.(2.5 (e) et (f))). Le paramètre β peut être choisi de la même manière que l'extraction des étages. Cependant, nous

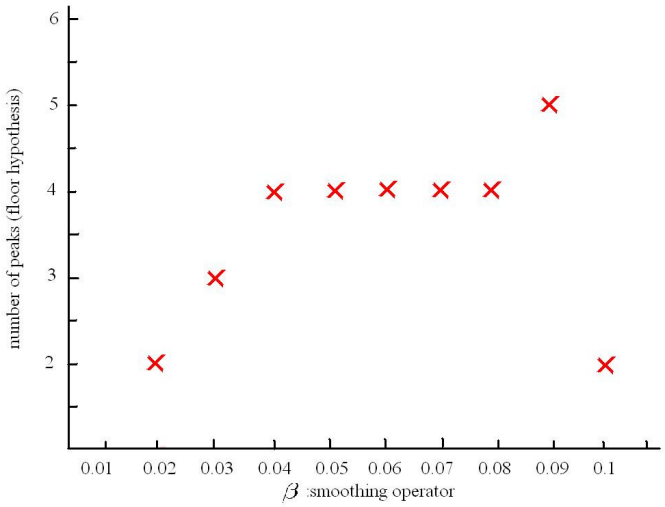


FIGURE 2.7 – **Evolution du nombre d'étages en fonction du paramètre de lissage β.**

notons que ce paramètre β reste régulièrement le même que celui déjà trouvé pour l'extraction des étages.

Pour résumer, l'algorithme d'extraction d'hypothèses de fenêtres est :

Algorithm 1 EXTRACTION AUTOMATIQUE D'HYPOTHESES DE FENETRES

ENTRÉE: Entrée : image rectifiée de la facade I_0

Initialisation : $\beta_0 \leftarrow 0.02$

Répète

1) Calcul des normes des composantes horizontales du gradient
2) Projection et accumulation des normes horizontalement
3) Evaluation du score S_{β_i} selon (eq.(2.1))
4) $\beta_i \leftarrow \beta_i + 0.01$

jusqu'à $\beta_i = 0.3$

Choisir $\beta_t = argmax_{\beta_i} \quad S_{\beta_i}$

Couper les étages avec β_t selon les "bosses" détectées sur l'histogramme seuillé par la moyenne de l'histogramme.

Calculer l'histogramme des normes des composantes verticales pour chaque étage avec β_t et chercher les "bosses" sur cet histogramme.

Les rectangles sélectionnés sont alors définis comme hypothèses de fenêtres.

2.1.3 Résultats

A présent, nous allons nous intéresser à l'évaluation de notre algorithme par rapport à l'algorithme de Lee et Nevatia [LN04] et par rapport au boosting. Afin d'évaluer les algorithmes, nous réalisons les expériences sur la base de données issue du camion Stéréopolis de l'IGN vu dans le chapitre 1.

2.1.3.1 La base de données

La base de données de l'IGN est constituée de clichés du 12 ème arrondissement de Paris. Pour chaque prise de position, nous avons 12 photos de l'emplacement. Cependant les 12 clichés ne sont pas tous intéressants dans notre cas d'étude. En effet comme nous le voyons sur la figure (Fig.(2.9)), les façades se situent sur les images prises par les caméras latérales. De plus les caméras de gauche (caméras 31 et 32 sur la figure (Fig.(2.9)) permettent de voir les facades de l'autres cotés de la rue et nous observons donc que la plupart des façades sont loin ou cachées par des véhicules. Par ailleurs, les clichés de la caméra 34 sont les devantures de boutiques et possèdent peu de fenêtres. Pour ces différentes raisons, nous nous sommes restreints aux clichés de la caméra 33. De plus, afin de se mettre dans les même conditions que l'état de l'art, nous redressons les images à l'aide d'un algorithme de redressement. Sur la figure (Fig.(2.8)), nous recherchons les points de fuites principaux, puis nous calculons ensuite l'homographie qui envoie ces points à l'infini.

(a) Image originale

(b) Recherche des droites convergentes vers les points de fuite principaux

(c) Image redressée, point de fuite envoyé vers l'infini

FIGURE 2.8 – **Redressement des images pour pouvoir comparer les algorithmes dans les conditions de l'état de l'art.**

Nous avons réalisé trois tests sur les façades de la rue Saint Antoine. La base est composée de 169 façades que nous avons préalablement redressées. Le premier test est la détection des fenêtres par la technique de boosting utilisée dans la thèse de Johansson [Joh02]. Le deuxième test est la détection à l'aide de l'algorithme de Lee et Nevatia [LN04]. Enfin nous avons comparé avec notre algorithme.

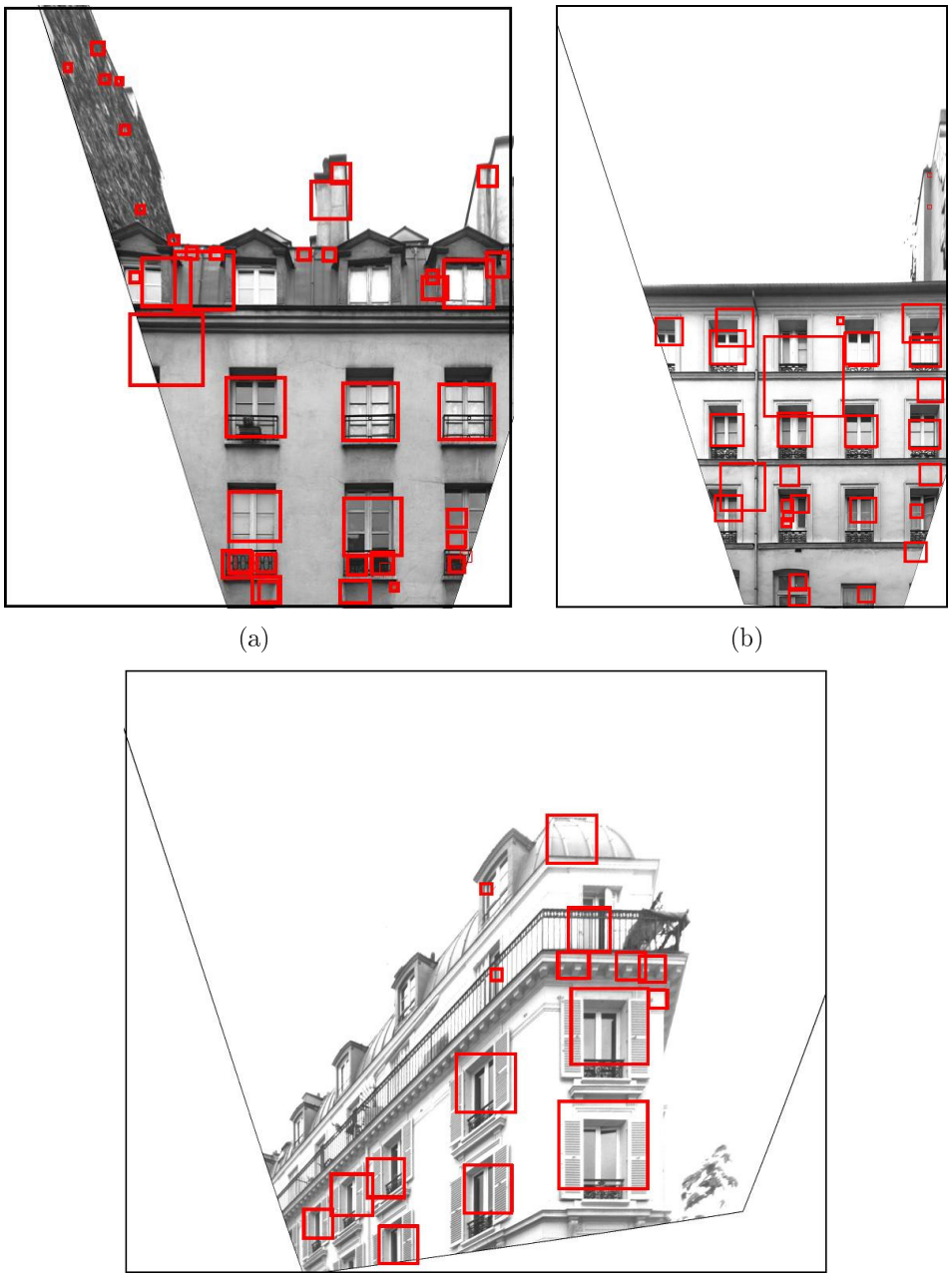
FIGURE 2.9 – **Clichés pris à un instant t par le camion Stéréopolis. Dans notre cas, seules les photos de la caméra 33 nous intéressent.**

2.1.3.2 Détection de fenêtres par boosting

Nous avons utilisé l'implémentation d'OpenCV (Open Source Computer Vision Library). Il s'agit d'une librairie de traitement d'images et de vision par ordinateur en langage C/C++. Cette bibliothèque propose un grand nombre d'opérateurs « classiques » : création, lecture et écriture d'images, accès aux pixels, traitement d'images, apprentissage, détection de visages, suivi d'objet vidéo, etc. Nous avons utilisé la détection d'objet à l'aide des descripteurs de Haar. Les descripteurs de Haar sont des fonctions permettant de connaître la différence de contraste entre plusieurs régions rectangulaires contiguës dans une image. Nous codons ainsi les contrastes existants dans une fenêtre et les relations spatiales.

Avec OpenCV, nous pouvons donc entraîner une cascade de plusieurs classifieurs en format xml pour la détection de fenêtre. Nous avons lancé l'apprentissage sur une base de 800 fenêtres et 2000 négatifs pour avoir une cascade avec un taux final de faux positifs inférieur à 0.5^{20}. Nous obtenons donc une cascade de 20 classifieurs forts avec chacun un taux de faux positifs inférieur à 0.5.

Selon les résultats du programme d'AdaBoost avec la cascade de classifieurs, cet algorithme est rapide pour détecter les fenêtres de différentes tailles et différentes luminosités. Cependant comme nous le constatons en regardant les résultats (Fig.(2.10)), de nombreux faux-positifs sont présents. Pour contrer cet inconvénient, il faut enrichir la base d'images de différents types de fenêtres pour entraîner des cascades plus pertinentes. Le nombre d'exemples d'apprentissage nécessaires pour bien apprendre en boosting est un des inconvénients de la méthode. Cela requiert beaucoup d'annotations, or nous souhaitons en utiliser peu.

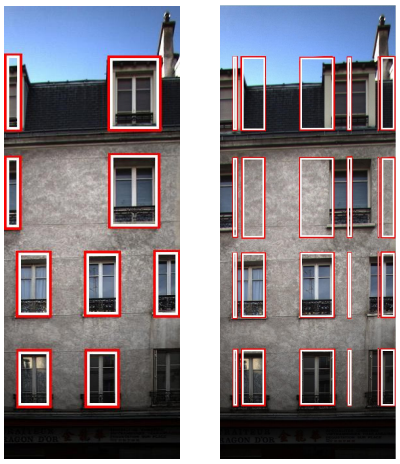


(a) (b) (c)

FIGURE 2.10 – **Extraction des hypothèses de fenêtres à l'aide d'AdaBoost. Cet algorithme détecte de nombreux faux-positifs.**

2.1.3.3 Comparaison de notre algorithme et de l'algorithme de Lee et Nevatia

Nous avons réalisé les expérimentations sur les 169 façades de la base. Quelques résultats sont montrés sur les figures (Fig.(2.11), Fig.(2.12), Fig.(2.13) et Fig.(2.14)).



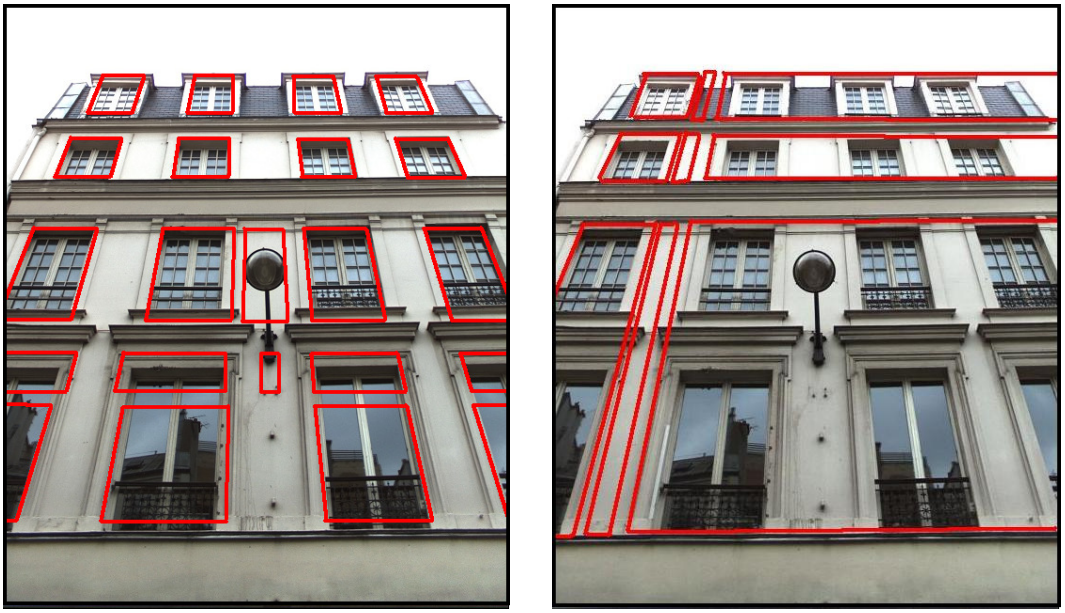
(a) (b)

FIGURE 2.11 – **Extraction des hypothèses de fenêtres. a)** Notre algorithme. **b)** L'algorithme de Lee et Nevatia. Nous constatons que l'alignement vertical considéré par Lee et Nevatia entraîne des faux positifs.

(a) (b)

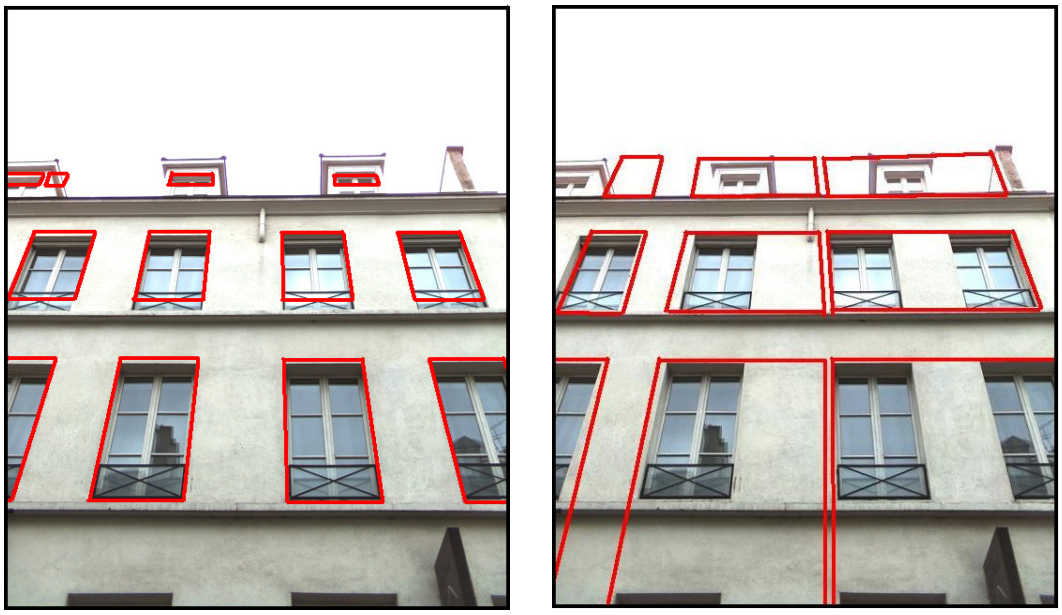
FIGURE 2.12 – **Extraction des hypothèses de fenêtres. a)** Notre algorithme. **b)** L'algorithme de Lee et Nevatia. Nous constatons qu'avec un mauvais paramètre de lissage, l'algorithme de Lee et Nevatia ne découpe pas correctement les étages.

En regardant les résultats nous constatons les différents point faibles de l'algorithme de Lee et Nevatia. En effet, l'alignement vertical des fenêtres n'est pas forcément vérifié et entraîne donc des faux positifs (Fig.(2.11)). De plus le problème d'avoir un paramètre non automatique dans l'algorithme de Lee et Nevatia contribue à un mauvais découpage des étages (Fig.(2.12), Fig.(2.13), Fig.(2.14)). En effet le paramètre de lissage et de dérivation n'étant pas forcément optimisé, l'histogramme d'accumulation des gradients n'est pas correctement défini et les étages proches sont alors détectés comme un seul étage. Au contraire, grâce à la recherche des étages, notre algorithme arrive à détecter les fenêtres non verticalement alignées. Et l'automatisation du meilleur paramètre entraîne un meilleur lissage et permet de lisser les textures entre des étages très proches et donc de mieux différencier les étages.



(a) (b)

FIGURE 2.13 – **Extraction des hypothèses de fenêtres. a)** Notre algorithme. **b)** L'algorithme de Lee et Nevatia. Nous constatons qu'avec un mauvais paramètre de lissage, l'algorithme de Lee et Nevatia est bruité par la texture du mur et réalise alors des groupements de fenêtres.



(a) (b)

FIGURE 2.14 – **Extraction des hypothèses de fenêtres. a)** Notre algorithme. **b)** L'algorithme de Lee et Nevatia. Nous constatons qu'avec un mauvais paramètre de lissage, l'algorithme de Lee et Nevatia ne découpe pas correctement les fenêtres. De plus l'alignement non vertical gêne l'algorithme.

Afin de comparer les deux algorithmes, nous avons évalué la détection de ces algorithmes par rapport à la vérité terrain des 169 façades (graphiques des figures (Fig.(2.15)) et (Fig.(2.16)). Dans toute la base, sur les 1825 fenêtres, nous constatons que notre algorithme en détecte 78% tandis que l'algorithme de Lee et Nevatia en détecte 45%. De plus, les deux algorithmes détectent beaucoup de fausse-alarmes. Cependant l'algorithme de Lee et Nevatia détecte deux fois plus de fausse-alarmes que notre algorithme.

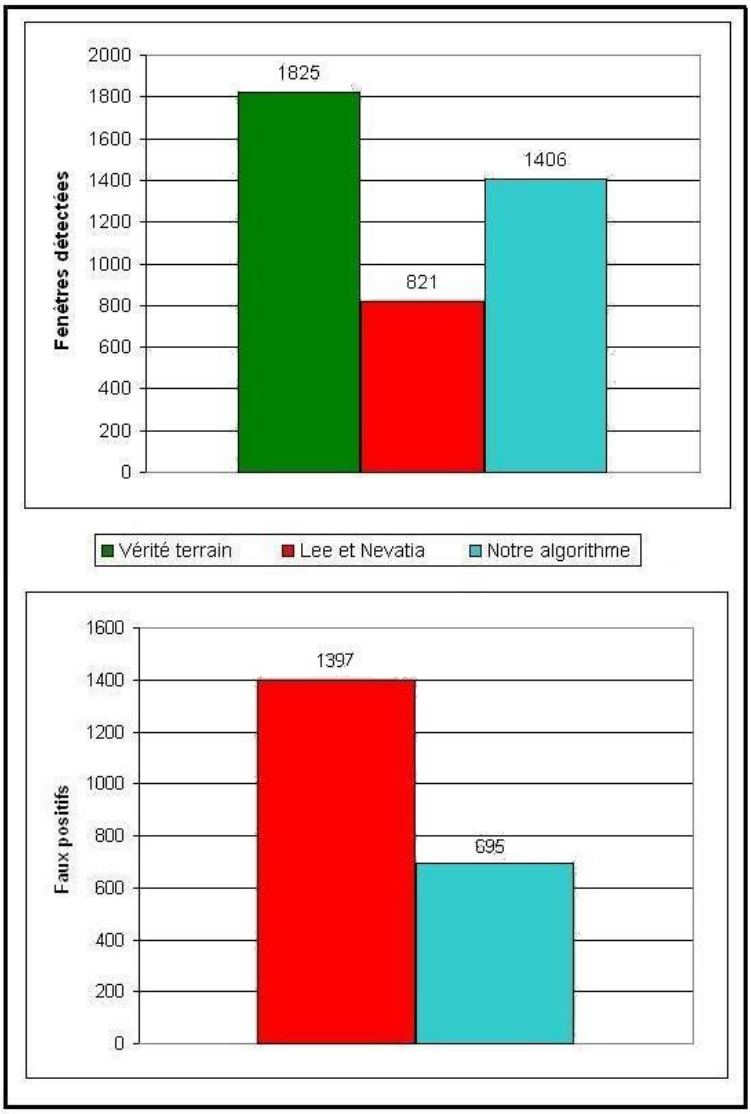


FIGURE 2.15 – **Evaluation des algorithmes de détection de fenêtres par rapport à la vérité terrain.**

Les premières observations sur les résultats de Lee et Nevatia étaient les regroupements des fenêtres (Fig.(2.16)) très nombreux à cause du mauvais paramètre de lissage et de dérivation. Sur le graphique de la figure (Fig.(2.16)), nous observons que l'algorithme de Lee et Nevatia effectue beaucoup de regroupements de 2 et 3 fenêtres. Ce graphique montre bien l'importance du choix du paramètre β.

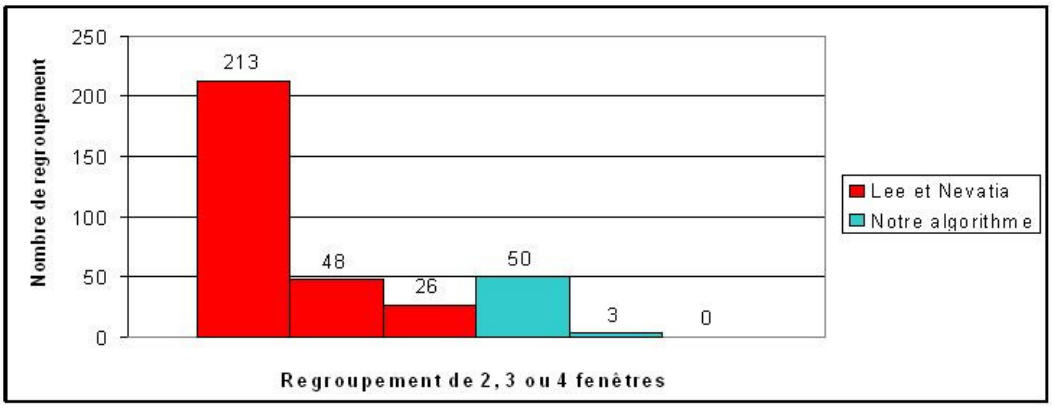


FIGURE 2.16 – **Evaluation des algorithmes de détection de fenêtres.** Le problème d'un mauvais lissage entraîne des regroupements de fenêtres.

2.1.3.4 Conclusion et limites de notre algorithme

Après un état de l'art sur la détection de fenêtres pour la modélisation des façades, nous avons proposé d'améliorer l'algorithme de Lee et Nevatia.

Notre algorithme s'appuie aussi sur l'alignement horizontal des fenêtres pour trouver les étages, puis ensuite découper les étages en fenêtres. Un point fort de notre algorithme est l'automatisation de la recherche du meilleur paramètre de lissage et dérivation. Les résultats montrent que nos améliorations permettent de trouver deux fois plus de fenêtres que celui de Lee et Nevatia avec moins de fausses alarmes.

Cependant, notre algorithme possède quelques limites. Comme la plupart des algorithmes, notre algorithme fonctionne principalement sur des façades vues de face. De plus, nous avons fait l'hypothèse que sur une image les fenêtres sont alignées horizontalement et forment donc des étages. Or certaines images (Fig.(2.17)) sont la jonction de deux façades et les étages ne sont pas forcément alignés.

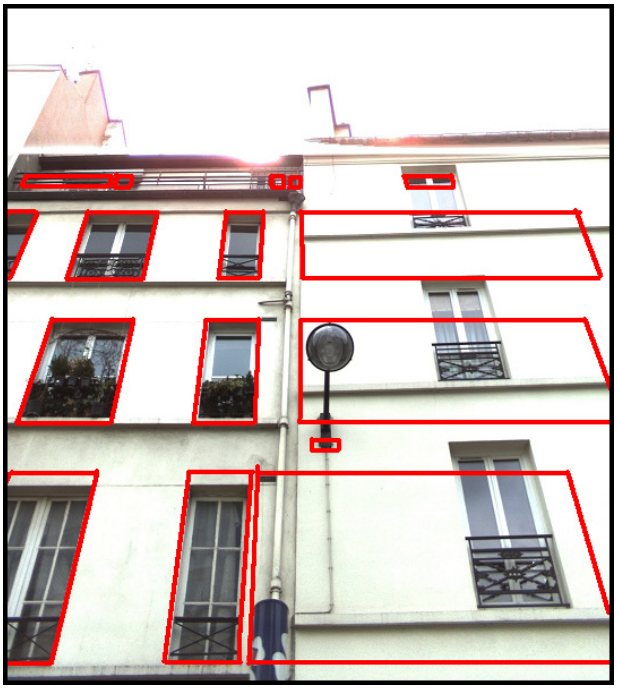


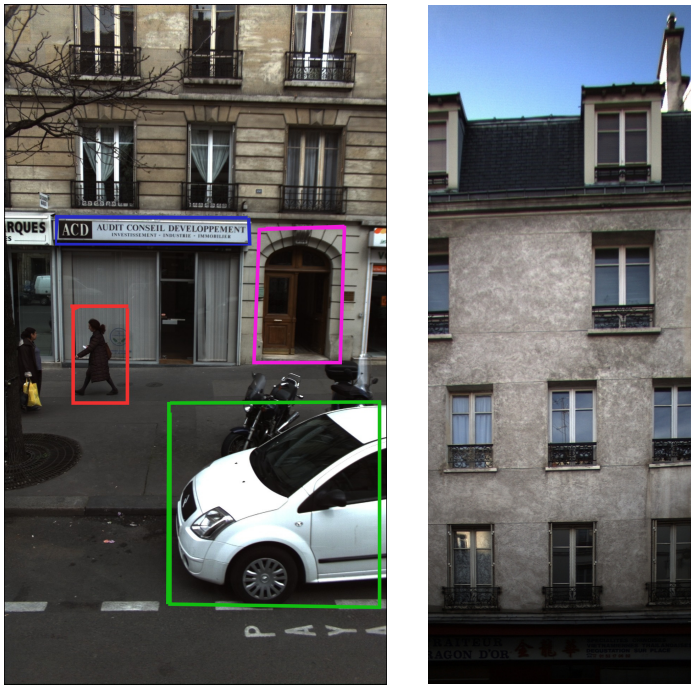
FIGURE 2.17 – **Problème d'extraction des fenêtres à la limite de deux façades.**

Par ailleurs, dans le choix de la base, nous avons sélectionné la caméra qui prend le haut des façades afin de ne pas être gêné par le rez-de-chaussée. Quelques tests sur les façades avec rez-de-chaussée montrent que notre algorithme est moins performant mais reste meilleur que le meilleur algorithme de l'état de l'art. En effet comme nous cherchons au préalable les étages avant de travailler sur chaque étage, nous ne sommes alors plus induits en erreur par les forts gradients du rez-de-chaussée (Fig.(2.18)).

(a)

(b)

FIGURE 2.18 – **Extraction des hypothèses de fenêtres. a)** Notre algorithme. **b)** L'algorithme de Lee et Nevatia. Nous constatons que la recherche des étages par notre algorithme permet de ne pas bruiter la recherche des fenêtres sur les autres étages.



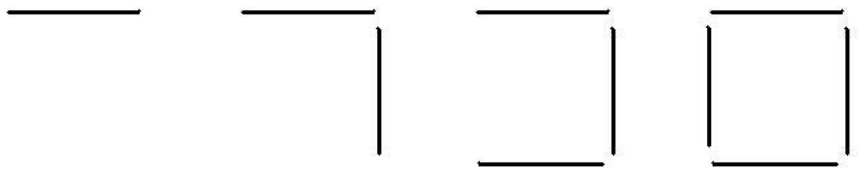
(a) (b)

FIGURE 2.19 – **Réponse de la question figure (Fig.(1.5)).** Notre système visuel est capable d'extraire sur : - l'image (a) un piéton, une voiture, du texte, une porte, et une façade avec plusieurs fenêtres. - l'image (b) une façade avec 4 étages et des fenêtres

2.2 Graphe Relationnel Attribué de contours

Dans le chapitre 1, à l'aide d'un simple test (Fig.(1.5) et Fig.(2.19)), nous avons confirmé l'intuition qu'avec uniquement les contours principaux nous arrivons à reconnaître les objets présents dans l'image. L'objectif dans la suite de cette thèse est de pouvoir structurer et comparer ces ensembles de contours. Comment décrire un objet à l'aide de ces contours ? Nous proposons d'utiliser un graphe pour apporter de la structure sur les ensembles de contours. Par exemple (Fig.(2.20)), un modèle simple de fenêtres peut être représenté par un premier contour horizontal puis un second contour vertical situé en bas à droite du premier puis un troisième contour horizontal en bas à gauche du second et enfin un quatrième contour vertical en haut à gauche. La signature de l'image sera donc basée sur un graphe dont les sommets et arcs sont valués. L'image est constituée de deux parties, d'une part des informations sur les contours (coordonnées du centre de gravité, longueur) et d'autre part les différentes relations topologiques entre contours. De récents travaux proposent plusieurs approches pour construire des

descripteurs basés contours dédiés à un objet spécifique. Shotton *et al.* [SBC05] et Opelt *et al.* [OPZ06] apprennent une distribution de contours de l'objet. Plus récemment, Ferrari *et al.* [FFJS08, FJS09] ont utilisé les propriétés de groupement perceptuel des contours en proposant un algorithme basé sur les parcours possibles dans une représentation réseau de l'image. Ils proposent une nouvelle méthode basée sur des paires de segments adjacents pour apprendre un modèle de forme d'un objet. Si nos travaux se sont simultanément orientés vers cette notion de groupement perceptuel de contours, dans nos travaux, nous structurons cette notion par un Graphe Relationnel Attribué.



(a) (b) (c) (d)

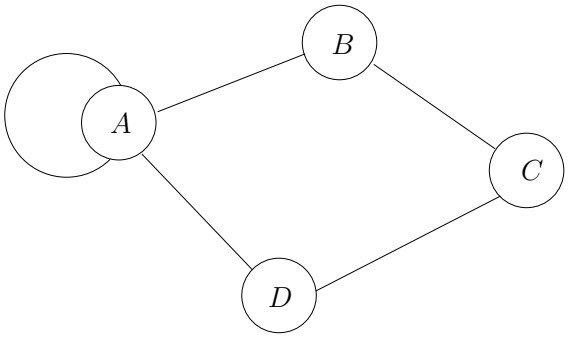
FIGURE 2.20 – **Modèle très simple d'une fenêtre !** Une fenêtre peut être représentée par un contour horizontal (a) puis d'un contour vertical situé en bas à droite (b) puis un contour horizontal en bas à gauche (c) et enfin un quatrième contour vertical en haut à gauche (d).

2.2.1 Définitions et notations

Avant d'introduire notre représentation d'image par un graphe de contours, nous présentons dans ce chapitre quelques définitions formelles de la théorie des graphes. Pour clarifier le vocabulaire de la théorie des graphes utilisé par la suite, nous avons donc choisi de garder le vocabulaire employé par C.Berge [Ber58].

Définition 2.1.

Graphe : *Un graphe (Fig.(2.21)) est un ensemble de points, dont certaines paires sont reliées par des lignes. Les points sont appelés sommets (ou noeud) et les lignes sont nommées arêtes. Plus formellement, un graphe* $G = (V, E)$ *est composé de deux ensembles, l'ensemble* E *des arêtes (edges) et l'ensemble* V *des sommets (vertices). L'ensemble des sommets est simplement une collection d'étiquettes qui permettent de distinguer un sommet d'un autre. L'ensemble des arêtes est constitué de paires non ordonnées d'étiquettes de sommets. Soit une arête* $e \in E$ *entre deux sommets* u *et* v $(u \in V, v \in V)$*, nous écrivons* $e = (u, v)$*.*



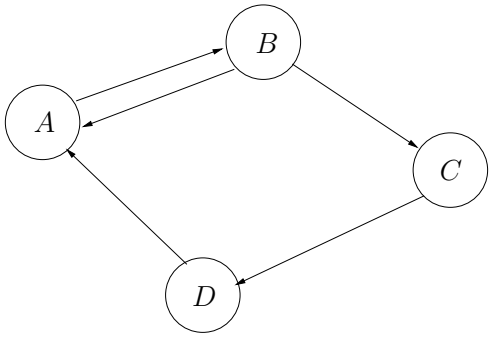
$V = \{A, B, C, D\}$: l'ensemble des sommets

$E = \{(A, B), (B, C), (C, D), (D, A), (A, A)\}$: l'ensemble des arêtes

FIGURE 2.21 – **Représentation d'un graphe et les ensembles définissant le graphe.**

Définition 2.2.

Graphe orienté (ou digraphe) : *En donnant un sens aux arêtes d'un graphe, on obtient un digraphe (ou graphe orienté). Une arête orientée d'un digraphe est appelée un arc. Un arc e de l'ensemble E est défini par une paire ordonnée de sommets. Lorsque* $e_{uv} = (u, v)$*, on dira que l'arc* e_{uv} *va de u à v. On dit aussi que u est l'extrémité initiale et v l'extrémité finale de* e_{uv}*.*



$V = \{A, B, C, D\}$: l'ensemble des sommets

$E = \{(A, B), (B, A)(B, C), (C, D), (D, A)\}$: l'ensemble des arcs

FIGURE 2.22 – **Représentation d'un graphe orienté et les ensembles définissant le graphe.**

Définition 2.3.

Graphe partiel (ou sous-graphe) : *Soit $G = (V, E)$ un graphe. Le graphe $G' = (V, E')$ est un graphe partiel de G, si E' est inclus dans E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G. Pour un sous-ensemble de sommets S inclus dans V, le sous-graphe de G induit par S est le graphe $G = (S, E(S))$ dont l'ensemble des sommets est S et l'ensemble des arêtes $E(S)$ est formé de toutes les arêtes de G ayant leurs deux extrémités dans S. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.*

Définition 2.4.

Graphe complet : *Dans un graphe complet, toutes les paires de sommets sont adjacentes. Un graphe complet à n sommets est noté Kn (le K est en l'honneur de Kuratowski 1966, un pionnier de la théorie des graphes).*

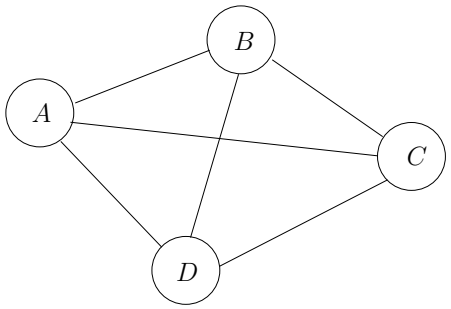


FIGURE 2.23 – **Graphe complet sur quatre sommets, noté K4.**

Définition 2.5.

Chaîne (marche, walk) : *Une chaîne dans un graphe est une séquence alternée de sommets et d'arêtes, commencant et se terminant par un sommet. La longueur d'une chaîne est le nombre d'arêtes utilisées.*

Définition 2.6.

Parcours ou chaîne simple (trail) : *Une chaîne simple dans un graphe est une séquence alternée de sommets et d'arêtes, où toutes les arêtes sont distinctes. La répétion de sommets est possible mais pas la répétition d'arêtes.*

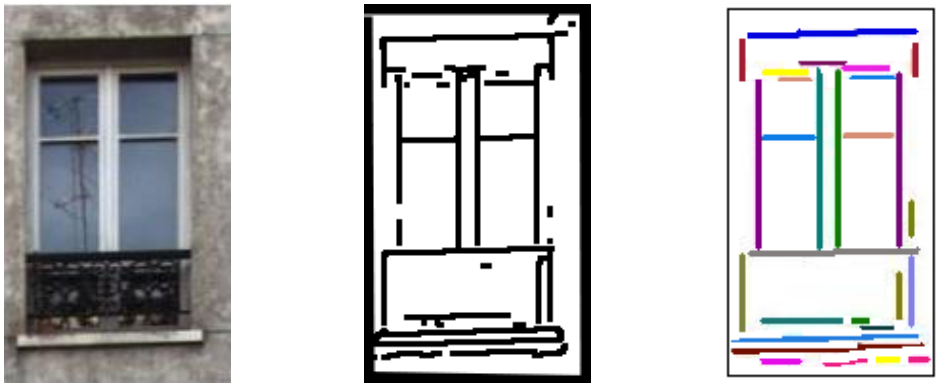
Définition 2.7.

Chemin path) : *Un chemin dans un graphe est une séquence alternée de sommets et d'arêtes où toutes les arêtes sont distinctes et les sommets sont distincts. Dans un digraphe, il s'agit d'une séquence d'arcs tous parcourus dans le même sens. Pour qu'un chemin relie deux sommets, un déplacement continu suivant une séquence d'arcs doit être possible.*

La longueur *du chemin est le nombre d'arcs utilisés, ou le nombre de sommets moins un.*

2.2.2 Notre représentation en graphe de contours

Pour chaque image, les segments de contours sont détectés à l'aide des opérateurs de Shen-Castan [SC92] (*cf.* paragraphe de la section 2.1.2), puis prolongés et polygonalisés (Fig.(2.24)).

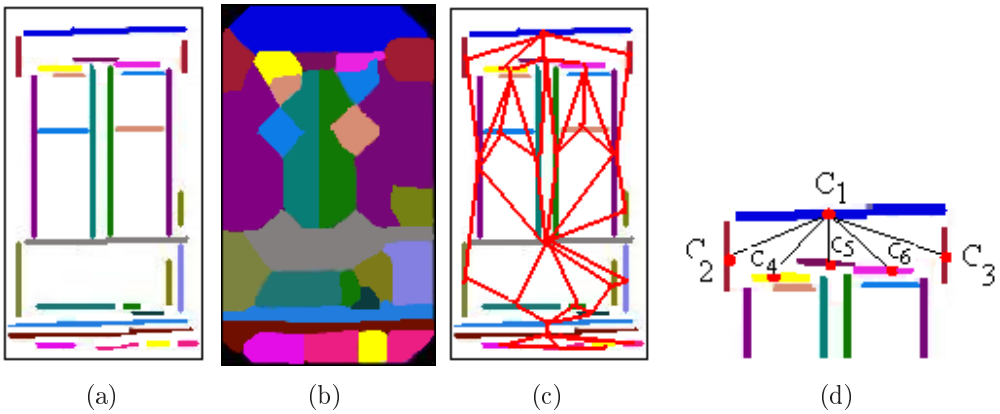


(a) (b) (c)

FIGURE 2.24 – **Segmentation : l'image est représentée par un graphe relationnel de segments de contours. (a)** Hypothèse de fenêtre. **(b)** Extraction des contours. **(c)** Polygonalisation.

Afin de considérer l'ensemble des segments comme une structure, nous représentons cet ensemble par un Graphe Relationnel Attribué (ARG). Chaque segment de contour C_i est représenté par un sommet v_i du graphe, l'arête e_{ij} du graphe représente elle la position relative entre deux segments de contours C_i C_j. L'information topologique (parallélisme, proximité) peut être considérée uniquement pour les voisins proches des segments de contours. Nous ne souhaitons pas utiliser un graphe complet pour des raisons de complexité de calcul et du peu d'informations que portent les arêtes entre des contours éloignés. La recherche des segments de contours les plus proches est réalisée à l'aide d'un diagramme de Voronoï [Aur91].

Définition 2.8. ***Diagramme de Voronoï :*** *Un diagramme de Voronoï représente une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace. Voronoï [Vor08] fut parmi les premiers à étudier et à décrire précisément les partitions spatiales. Il formalisait l'idée intuitive de diviser l'espace en considérant un ensemble fini de points fixés et en associant chaque point de l'espace au point le plus proche. Les régions définies par cette construction sont nommées régions de Voronoï. Sur la figure (Fig.2.25 (b)), nous utilisons le diagramme de Voronoï en associant chaque point de l'espace au segment de contours le plus proche.*



(a) (b) (c) (d)

FIGURE 2.25 – **Extraction et graphe de segments de contours. (a)** Segments de contours extraits et polygonalisés. **(b)** Diagramme de Voronoi sur les segments de contours. **(c)** Graphe représenté en rouge **(d)** Zoom sur une partie du graphe. C_1 est relié par une arête á C_2,C_3,C_4,C_5,C_6, mais pas aux autres segments non adjacents dans le diagramme de Voronoï.

Une arête dans le graphe relationnel représente la relation entre deux segments proches dans le sens où leurs cellules de Voronoï sont adjacentes.

Nous obtenons un graphe de contours. Afin d'améliorer la description du graphe et pouvoir ensuite comparer les graphes, nous allons valuer ce graphe. L'information des contours sera portée par les sommets et l'information structurelle par les arêtes.

2.2.2.1 Attributs de sommet

Afin d'être robuste aux changements d'échelle, un segment de contour est uniquement caractérisé par son orientation (horizontale, verticale...). Un segment C_i dépend donc de l'angle Θ formé avec l'axe horizontal (mesuré en degrés). Sur la figure (Fig.(2.26 (a))), nous notons que $\Theta \in [0, 180[$. Une fonction de distance

ou mesure de similarité est alors à définir pour pouvoir comparer les segments entre eux. Dans quelle mesure les angles Θ_i ainsi constitués sont-ils exploitables afin d'évaluer l'orientation des segments ? Un segment d'angle π est-il plus proche d'un segment d'angle 0 que d'un segment d'angle $\pi/2$? Afin d'évaluer et de comparer l'orientation des segments, nous proposons de représenter les segments C_i par des vecteurs v_i sur le cercle trigonométrique (Fig.(2.26)).

Etant donné que l'on caractérise un segment par son orientation, le segment C_i est alors représenté dans le graphe par le sommet $v_i = (cos(2\Theta), sin(2\Theta))^T$. Nous avons pris 2Θ afin de faire la différence entre les contours quasi-parallèles et les contours perpendiculaires à l'aide du produit scalaire.

En effet, prenons un exemple :

Soient trois contours C_1, C_2, C_3, tel que C_1 soit perpendiculaire à C_2 et parallèle à C_3.
C_1 forme un angle Θ_1 avec l'axe horizontal.
C_2 forme un angle $\Theta_2 = \Theta_1 + \frac{\pi}{2}$.
C_3 forme un angle $\Theta_3 = \Theta_1 + \pi$.

Si nous caractérisons C_i par $v_i = (cos(\Theta_i), sin(\Theta_i))^T$, et calculons les différents produits scalaires des vecteurs, nous avons :

$$\begin{aligned} < v_1, v_2 > &= cos(\Theta_1)cos(\Theta_2) + sin(\Theta_1)sin(\Theta_2) \\ &= cos(\Theta_1 - \Theta_2) \\ &= cos(\Theta_1 - \Theta_1 - \frac{\pi}{2}) \\ &= 0 \end{aligned}$$

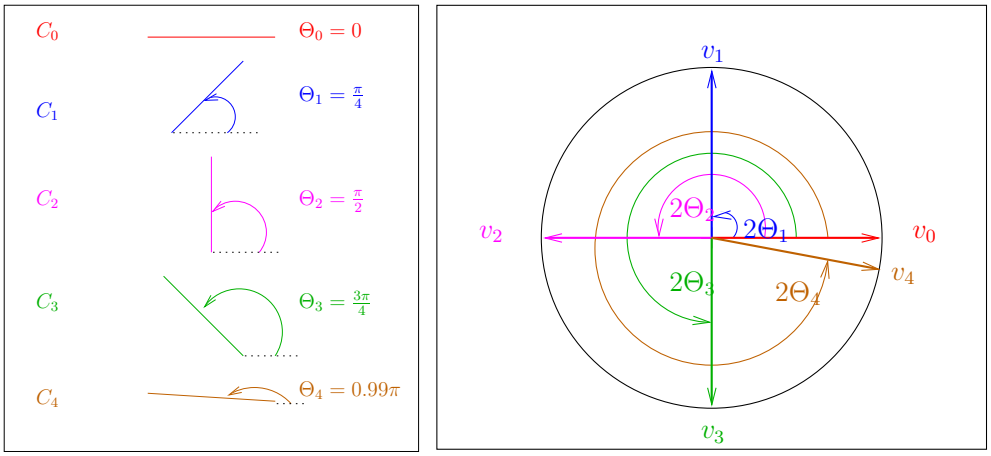
De même : $< v_1, v_3 >= -1$,$< v_2, v_3 >= 0$.

Nous avons donc C_1 qui est plus similaire à C_2 qu'à C_3, ce qui est absurde dans notre cas.

Par contre, en posant $v_i = (cos(2\Theta_i), sin(2\Theta_i))^T$, nous obtenons :

$$\begin{aligned} < v_1, v_2 > &= -1 \\ < v_2, v_3 > &= -1 \\ < v_1, v_3 > &= 1 \end{aligned}$$

Ainsi comme le montrent la figure (Fig.(2.26)) et le tableau (Tab.(2.1)), le produit scalaire entre la représentation vectorielle de deux contours parallèles (vecteurs colinéaires non opposés) est proche de 1 alors que le produit scalaire entre la représentation vectorielle de deux contours perpendiculaires (vecteurs colinéaires et opposés) est -1.



(a) (b)

FIGURE 2.26 – **Représentation des segments de contours en fonction de leur orientation. (a)** Les segments de contours forment un angle $\Theta \in [0, 180[$ avec l'axe horizontal. **(b)** Les segments sont représentés chacun par un vecteur unitaire $v_i = (cos(2\Theta), sin(2\Theta))^T$

$< ., . >$	v_0	v_1	v_2	v_3	v_4
v_0	1	0	-1	0	0.93
v_1	0	1	0	-1	-0.06
v_2	-1	0	1	0	-0.93
v_3	0	-1	0	1	0.06
v_4	0.93	-0.06	-0.93	0.06	1

TABLE 2.1 – Tableau exemple des produits scalaires des vecteur v_i

2.2.2.2 Attributs d'arête, Relations topologiques

Le concept de topologie, bien qu'il nous paraisse abstrait, se rapporte à une démarche courante de notre esprit pour appréhender la réalité. Notre perception visuelle est topologique ! Lorsque nous observons un paysage, un lieu, ou encore lorsque nous consultons une carte, un plan cadastral, notre perception immédiate est globale. Les objets tels que bâtiment, portion de forêt, une agglomération sont "vus" dans leur contexte. La notion de voisinage est implicite : la route traverse l'agglomération, la boulangerie est situé à côté de la boucherie. Au sens de notre appréhension de l'espace géographique, la topologie est donc l'ensemble des relations perçues qui nous permettent de situer les objets les uns par rapport aux autres. Le "voisinage" est donc une notion spatiale "qu'est-ce qui est à côté de quoi ?" Pour les réseaux, la question devient "qu'est-ce qui est connecté à ?" La notion de topologie est un élément fondamental de l'analyse structurelle. C'est une notion interdisciplinaire qui a attiré l'intérêt des différentes communautés scientifiques, non seulement en informatique, mais en linguistique (Lautenschütz *et al.* [LDR+07]), philosophie (Casati et Varzi [CV99]), et psychologie (Ishikawa *et al.* [IM06]). En informatique, différents domaines, tels que les bases de données spatiales (Güting [GÖ4]), les systèmes d'information géographique (SIG) (Egenhofer et Mark [EM95]), les bases de données d'images (Berretti *et al.* [BDBV03]), et le raisonnement spatial qualitatif (Freksa [Fre92], Hazarika et Cohn [HC01]) impliquent la recherche sur les relations spatiales. Les mathématiques en donnent une définition rigoureuse "Propriétés des êtres géométriques subsistant après une déformation continue, et qui fait abstraction de la notion de distance". Elle est parfois appelée de manière raccourcie : une géométrie sans métrique. Pour les autres disciplines, le sens est plus large. En sciences humaines, la topologie signifie un arrangement, une configuration d'un groupe de notions et de leurs relations.

Dans le cadre du traitement d'images, les relations topologiques sont couramment utilisées pour représenter les relations entre les différentes entités présentes dans l'image. La représentation des relations entre régions de l'image est l'une des utilisations les plus courantes. Ainsi il existe différents moyens de représenter les relations spatiales entre régions. Malki *et al.* [MZM02] emploient les relations d'Allen [All83] pour définir des relations topologiques et d'orientation (telles que chevauchement, recouvrement, disjonction) entre objets. Des attributs flous de positions relatives ont été proposés par Bloch *et al.* [Blo99] pour des ensembles nets et par Krishnapuram *et al.* [KKM93] pour des ensembles flous. Dans [PFVL05]et [PFGG09] , Philipp *et al.* proposent d'utiliser la position relative des régions floues formant un objet. Par exemple dans une voiture vue de profil, les roues sont situées sous la carrosserie, les vitres en haut et le tout se trouve en général sur une zone de macadam. Les auteurs ont défini une matrice binaire d'adjacence des régions floues : l'adjacence vaut 1 si les deux régions ont

au moins un pixel en commun, 0 sinon et ils représente l'information spatiale par un vecteur entre les centres de gravité des régions floues. Lebrun *et al.* [LPFG08] proposent de valuer les adjacences et proposent une description composée de 4 éléments pour représenter la position spatiale. Une arête est donc un lien orienté entre deux régions. Elle est décrite par les 4 éléments : dessus, dessous, gauche et droite. Chaque élément représente la "granularité" entre deux régions connexes. Ainsi, pour deux régions R_i et R_j, les auteurs proposent de valuer l'élément gauche de R_j par la proportion du nombre de pixels de R_j ayant un pixel voisin de R_i à sa gauche.

Dans notre cas la structure de l'objet (la relation spatiale entre les contours) est représentée par un arc du graphe $e_{ij} = (v_i, v_j)$. Cet arc représente la proximité entre le segment de contour C_i et C_j (si C_i et C_j ne sont pas proches au sens de Voronoï, e_{ij} n'existe pas). L'arête dans le graphe est caractérisée par la position relative des centres de gravité des segments de contours C_i et C_j, notés $g_{C_i}(Xg_{C_i}, Yg_{C_i})$ et $g_{C_j}(Xg_{C_j}, Yg_{C_j})$. L'arête est donc affectée d'un vecteur $e_{ij} = (Xg_{C_j} - Xg_{C_i}, Yg_{C_j} - Yg_{C_i})^T$.

2.3 Conclusion

Dans cette section, nous avons extrait des hypothèses de fenêtre à l'aide d'un algorithme automatisé qui s'appuie sur les propriétés des fenêtres. Nous avons apporté deux contributions à cet algorithme inspiré d'un algorithme de l'état de l'art (Lee et Nevatia [LN04]). La première contribution est la recherche des étages puis des fenêtres par étages. En effet cet algorithme découpe les étages à partir d'un histogramme des composantes horizontales du gradient, puis il découpe les étages en hypothèses de fenêtres à partir de l'histogramme des composantes verticales du gradient. La deuxième contribution est l'automatisation de la recherche du meilleur facteur d'échelle. Ainsi nous cherchons le meilleur paramètre de lissage et dérivation qui permet de garder les gradients principaux des fenêtres et d'écarter le bruit.

Nous avons ensuite représenté les images sous la forme d'un graphe d'adjacence de segments de contours, valué par des informations d'orientation et de proximité des segments. Nous nous plaçons donc dans un contexte d'appariement de graphes. Afin de classer ces graphes d'objets, nous allons maintenant définir une similarité capable de comparer ces graphes.

Chapitre 3

Etat de l'art : appariement de graphes

L'utilisation de graphes est très courante dans de nombreux domaines comme le soulignent Conte et al. dans [CFSV04]. Les graphes sont utilisés en reconnaissance de caractères et de nombres (Filatov et al. [FGK95], Suganthan et al. [SY98]). Dans le domaine de la bio-informatique et chimie, les graphes permettent de manipuler des représentations moléculaires (Fischer et al.[FGB09], Kashima et al. [KT04]). Dans le cadre de l'analyse de textes (Amghar [ABC01]), les graphes permettent de représenter les documents afin de mettre en évidence l'agencement spatial d'un document. Comme nous pouvons le constater, le but principal des graphes est de permettre une représentation structurée des données. Il est ainsi possible de représenter et de manipuler plus aisément des objets complexes.

Dans le chapitre 2, nous avons vu qu'avec uniquement l'information des contours, l'oeil humain arrive à reconnaitre les différents objets. Ainsi pour comparer différents objets entre eux, nous proposons de les comparer en utilisant leurs contours. Les contours permettent de représenter la forme de l'objet (shape context de Belongie *et al.* [BMP00]). Afin de comparer les ensembles de contours entre eux et en prenant en compte leur structure, nous les transformons en graphes valués. Nous proposons d'aborder les graphes comme outil de représentation des objets. Notre problème revient alors à pouvoir classer et discriminer les graphes. Le problème de la classification des objets peut être considéré comme un problème d'appariement inexact de graphes (Emms et al.[EWH09], Shokoufandeh et al. [SBM+06]). Le problème est double : tout d'abord trouver une mesure de similarité entre 2 graphes de tailles différentes et d'autre part trouver le meilleur appariement entre 2 graphes dans un temps "acceptable".

Dans ce chapitre, nous présentons différentes techniques d'appariement de graphes. Parmi toutes ces techniques, nous avons choisi de nous intéresser tout particulièrement à concevoir une similarité par fonction noyau, qui offre de nom-

breux avantages pour l'analyse et l'exploitation des graphes dans le cadre de l'apprentissage. Nous voyons ensuite les noyaux, leurs constructions et leurs propriétés. Puis nous parlons des noyaux sur graphes et plus particulièrement des noyaux définis par Kashima *et al.* [KT04] et Lebrun *et al.* [LPFG08]. Nous proposons ensuite nos noyaux sur graphes adaptés au calcul de similarité sur les contours. Enfin nous voyons quelles sont les différents contextes de classification utilisées à l'aide des nouvelles similarités sur graphes de contours. En fonction de l'application, soit nous choisissons une fonction de similarité symétrique utilisée comme un noyau dans un classifieur Séparateur à Vaste Marge, soit nous définissons une similarité non symétrique basée sur un classifieur des plus proches voisins.

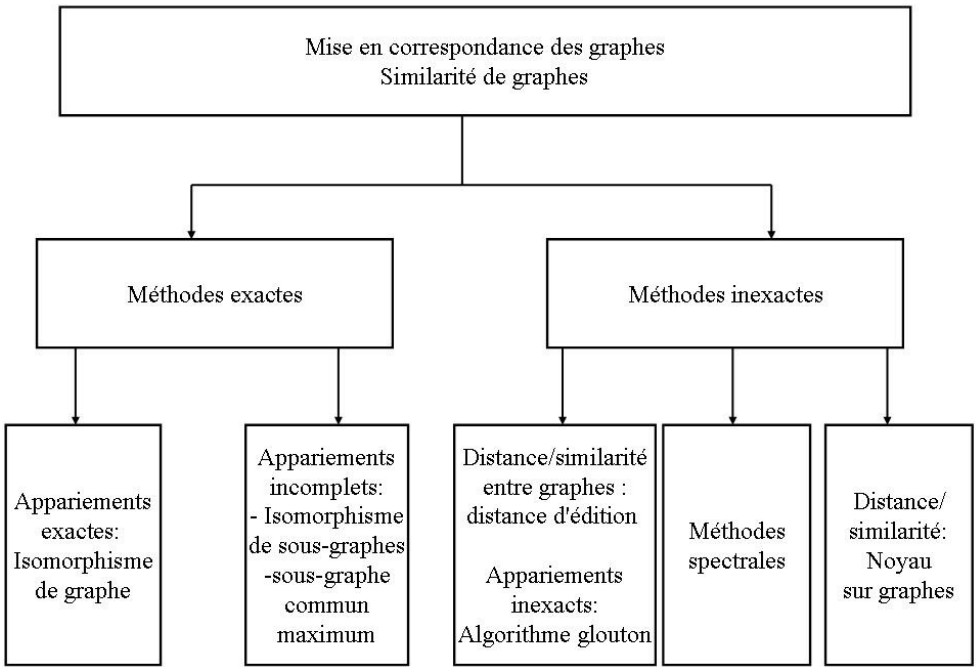


FIGURE 3.1 – Reconnaisance d'objets à base de graphes.

3.1 Appariement de graphe (graph matching)

Pour calculer la similarité entre graphes, il est possible d'utiliser un algorithme de *graph matching* (Torsello [Tor04]), littéralement effectuer la mise en correspondance de graphes. Cette approche consiste à évaluer les différences entre les graphes au niveau des sommets et des arcs afin de pouvoir comparer les graphes entre eux.

Il existe deux types de mise en correspondance de graphes (Fig.(3.1)) :

- la correspondance exacte
- la correspondance inexacte

3.1.1 Méthodes exactes

La correspondance exacte est fondée sur le principe de l'isomorphisme de graphe. Le but consiste à chercher une bijection d'un graphe à l'autre, sans modi-

fier leur structure. La principale contrainte réside dans la conservation des liaisons entre les sommets.

Soient 2 graphes : $G = (V, E)$ et $G' = (V', E')$ avec $|V| = |V'|$, une application $f : G \rightarrow G'$ est un morphisme de graphe si $f = (f_v, f_e)$ où $f_v : V \rightarrow V'$ transforme les sommets de G en ceux de G' et $f_e : E \rightarrow E'$ les arêtes de G en celles de G' tel que si une arête $e_{ij} \in E$ entre deux sommets de G existe alors $e_{f_v(i)f_v(j)} \in E'$ entre deux sommets de G' doit exister. Si f_v et f_e sont bijectives alors f est un isomorphisme.

Une comparaison moins contraignante consiste à rechercher les isomorphismes entre les sous-graphes des deux graphes. Les méthodes de comparaison exacte sont nombreuses et la plupart sont basées sur une représentation par arbre de recherche. Chaque mise en correspondance est représentée sous la forme d'une chaîne dans un arbre de recherche. Ainsi chaque mise en correspondance entre deux sommets est représentée par un noeud dans l'arbre de recherche, et chaque mise en correspondance entre deux arêtes est mise sous la forme d'un lien entre deux noeuds de l'arbre de recherche.

Parmi les méthodes de comparaison exacte, nous pouvons citer la recherche d'isomorphisme à partir de matrice d'adjacence (Shearer et al.[SBV01]) et la recherche du plus grand sous graphe commun à partir d'arbre de décision (Messmer [Mes95], Ullman [Ull76], Bunke [BS98]). Cordella et al. [CFSV01] proposent d'utiliser des heuristiques plus efficaces afin de traiter des graphes de plus grande taille.

Ces différents algorithmes sont NP-complets et ont donc des temps de résolution exponentiels. Cependant sur des petits graphes de quelques sommets avec des attributs symboliques, les algorithmes classiques fonctionnent en un temps raisonnable. Dans notre cas, ces algorithmes ne conviennent pas du fait de la nature de nos graphes qui sont de grande taille et dont les sommets et arêtes sont caractérisés par des vecteurs. De plus, la recherche d'isomorphisme dans les graphes est une contrainte de mise en correspondance trop forte dans le contexte de similarité d'image. En effet, la notion de similarité d'image est associée au compromis à trouver entre le pouvoir discriminant (des graphes différents représentent des objets différents) et le potentiel de généralisation (des graphes différents peuvent représenter le même objet vu différemment).

Dans cette thèse, nous nous intéressons à la classification de graphe, c'est à dire à quel point un graphe G est plus similaire à un graphe G' qu'à un graphe G''.

3.1.2 Méthodes inexactes

La correspondance inexacte permet de mettre en correspondance des graphes dont la topologie est différente. Les méthodes les plus anciennes se sont inspirées des méthodes exactes. L'approche de Messmer [Mes95] sur la décomposition en sous-problème pour la recherche du plus grand sous-graphe commun a été étendue pour permettre une certaine erreur dans la mise en correspondance (Messmer et Bunke [MB98]). Nous présentons ci-dessous d'autres méthodes inexactes de comparaison de graphes.

3.1.2.1 Distance d'édition

Dans l'article de Le Saux et Bunke [LSB06], la distance entre graphes est mesurée à l'aide de la méthode basée sur la "distance d'édition". La distance est obtenue par le coût de transformation d'un graphe à l'autre. Le principe consiste à appliquer différentes opérations sur les sommets et les arcs d'un graphe G afin que celui-ci corresponde à un graphe G'. Il est ainsi possible de supprimer ou de fusionner des arcs et des sommets ou encore d'en insérer de nouveaux. La similarité est évaluée par le nombre et l'importance des transformations. Le score obtenu pour la transformation d'un graphe en un autre est ainsi comparé avec les scores obtenus pour les transformations en d'autres graphes (Le score le plus faible correspondra au graphe le plus proche). L'inconvénient majeur de cette solution réside dans la non symétrie de la solution (Le Saux et Bunke [LSB06]). En effet, les transformations pour transiter d'un graphe vers un autre ne sont pas forcément symétriques, et les coûts associés à ces transformations ne sont donc pas les mêmes que leurs réciproques.

3.1.2.2 Méthodes spectrales

Les méthodes spectrales s'appuient sur les vecteurs et valeurs propres de la matrice d'adjacence des sommets pour comparer les graphes (Zhu et Wilson [ZW08]). L'intérêt d'une telle approche repose sur le fait que les vecteurs et valeurs propres sont invariants aux permutations des sommets du graphe. Ainsi, deux graphes isomorphes ont le même spectre. Notons toutefois que la réciproque n'est pas vraie, deux graphes de même spectre ne sont pas nécessairement isomorphes. En terme de complexité cette approche est intéressante puisque le principal calcul réside dans la détermination du spectre, qui peut être pré-calculé. Les premières méthodes basées sur ce principe sont assez contraintes, par exemple la méthode de Umeyama [Ume88] ne fonctionne que sur des graphes de même nombre de sommets et tous les sommets doivent être mis en correspondance deux à deux. Des méthodes ont été proposées par la suite pour réduire ces problèmes (Xu *et al.* [XK01]). Une autre catégorie de méthodes s'appuie sur le spectre pour effectuer

un clustering des sommets. Par exemple, Carcassoni *et al.* [CH01] utilise ces clusters pour effecteur une mise en correspondance hiérarchique. Un premier niveau compare les clusters, puis une mise en correspondance plus fine est effectuée au sein des clusters. La méthode de Kosinov *et al.* [KC02] construit un espace vectoriel défini par les vecteurs propres de la matrice d'adjacence, puis projette les sommets dans cet espace. Un clustering est alors utilisé pour trouver les sommets à mettre en correspondance.

3.1.2.3 Méthodes à noyaux

Des méthodes récentes (Gärtner [Gär03], Vishwanathan [VBKS08]) propoposent de projeter les graphes dans un espace vectoriel afin de pouvoir utiliser des métriques et distances adaptées aux vecteurs. Les auteurs se placent alors dans le cadre des méthodes à noyaux. Les noyaux sur graphes correspondent à un produit scalaire dans un espace implicite entre des vecteurs décrivant les graphes. En changeant d'espace, nous pouvons utiliser des méthodes travaillant directement dans l'espace des graphes mais aussi bénéficier de l'ensemble des avantages des méthodes travaillant dans l'espace induit des descripteurs. Parmi ces méthodes, nous pouvons citer les méthodes basées sur les chemins aléatoires (Kashima et Tsuboi [KT04], Suard et al. [SGRB05]) qui se focalisent sur des chemins issus des graphes.

En pratique l'intégralité des chemins n'est pas considérée, et diverses approches sont proposées pour choisir le meilleur tirage. Par exemple, nous pouvons considérer que les chemins issus de chemins aléatoires.

3.1.3 Discussions

De nombreuses méthodes exactes de comparaison de graphes s'intéressent à la structure du graphe en recherchant des isomorphismes. Ces techniques sont régulièrement employées en chimie où les molécules sont représentées par des graphes. Il s'agit de petits graphes qui portent peu d'information, les sommets sont des symboles (carbone, oxygène...) et les arêtes sont étiquetées par "à une liaison avec" ou "est au dessus de". Dans le cadre de la classification d'images, les sommets et arêtes sont représentés par des vecteurs. Il devient alors difficile d'évaluer si un sommet est identique à un autre. Les méthodes inexactes, basées sur des contraintes plus souples et le calcul de similarités, semblent donc plus appropriées à notre contexte. L'objectif n'est pas d'avoir une comparaison exacte mais une valeur de similarité entre graphes qui permet de les classer.

Par ailleurs nous souhaitons réaliser un apprentissage afin de classer nos graphes. Les approches basées noyaux sont alors pertinentes. Elles permettent de

passer dans un espace vectoriel décrivant les graphes et offre un cadre mathématique pour l'apprentissage. Dans le cas des graphes, nous allons utiliser des noyaux de graphes calculés sur des chemins aléatoires. Le but consiste à extraire un ensemble de chemins de chaque graphe puis de comparer les chemins entre eux et combiner les comparaisons pour obtenir le noyau final. La définition d'un noyau de graphe permet de bénéficier de la théorie sous-jacente des méthodes à noyaux. Avant de présenter les noyaux sur graphes et nos contributions sur de nouveaux noyaux sur graphes, nous allons introduire les méthodes à noyaux.

3.2 Les noyaux

3.2.1 Noyaux de Mercer

Dans cette thèse, nous avons choisi d'utiliser une similarité basée sur des fonctions noyaux. Cette approche ne fournit pas une métrique particulière pour tels ou tels types de signatures, mais un cadre formel qui offre de nombreux avantages (Vapnik [Vap98], Smola *et al.* [SBSS00] et Shawe-Taylor et Cristianini [STC04]). Le premier objectif de ce cadre est de déplacer le problème initialement exprimé dans un espace quelconque $\mathcal{X}$ (dans notre cas, l'espace des graphes) dans un espace hilbertien $\mathcal{H}$. L'idée derrière ce choix est de ramener tout espace à un espace vectoriel bien connu, et muni de métriques elles aussi bien connues : le produit scalaire et la distance Euclidienne.

Le changement d'espace se réalise grâce à une fonction $\Phi : X \in \mathbb{R}^m \rightarrow \Phi(X) \in \mathbb{R}^p$ Ce nouvel espace a souvent une dimension p plus grande que la dimension m de l'espace initial. Cet espace $\Phi(X)$ est appelé "espace de redescription". Ainsi, intuitivement, plus la dimension de l'espace de redescription est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée.

Toutefois, cette méthodologie repose sur une bonne définition de la fonction Φ :

- lorsque l'espace de description est de haute dimension, il peut être calculatoirement coûteux d'effectuer le passage vers l'espace de description avant d'effectuer le calcul du produit scalaire. Une expression optimisée du produit scalaire sans transformation explicite peut être plus adaptée.

- dans le cas d'un espace de description de dimension infini comme celui induit par une fonction gaussienne, la définition explicite de Φ est impossible même si son existence est prouvée.

Il n'est donc pas commode de chercher à définir directement la fonction Φ. Dans le cas des classifieurs à hyperplan, heureusement, seul le produit scalaire entre les éléments intervient pour définir l'hyperplan. C'est ce qu'on appelle l'astuce du noyau (kernel trick) qui consiste à ne travailler que sur les valeurs du produit scalaire dans l'espace induit, soit la fonction K associée à la fonction Φ :

$$K(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >$$

Nous verrons dans la section suivante comment définir un noyau sans pour

autant définir explicitement Φ. Afin d'introduire la notion de noyau, nous donnons quelques définitions.

3.2.2 Définitions

Définition 3.1.

Produit Scalaire : *Soit $\mathcal{E}$ un espace vectoriel, l'application $k : \mathcal{E} \times \mathcal{E}$ est un produit scalaire dans $\mathcal{E}$, si k est une application symétrique bilinéaire strictement positive.*

k vérifie donc les propriétés suivantes :

- Symétrique : $\forall(\vec{x}, \vec{y}) \in \mathcal{E}^2, k(\vec{x}, \vec{y}) = k(\vec{y}, \vec{x})$
- Bilinéaire : $\forall(\vec{x}, \vec{y}, \vec{z}) \in \mathcal{E}^3, k(\vec{x} + \vec{z}, \vec{y}) = k(\vec{x}, \vec{y} + \vec{z})$
  et $\forall\alpha \in \mathcal{R}, k(\alpha\vec{x}, \vec{y}) = k(\vec{x}, \alpha\vec{y}) = \alpha k(\vec{x}, \vec{y})$
- Définie positive : $\forall\vec{x} \in \mathcal{E} \setminus \{0\}, k(\vec{x}, \vec{x}) > 0$
  $(k(\vec{x}, \vec{x}) = 0 \Leftrightarrow \vec{x} = \vec{0})$.

Définition 3.2.

Un espace de Hilbert : *Un espace de Hilbert $\mathcal{H}$ est un espace vectoriel complet doté du produit scalaire $<.,.>$, dont la norme est associée à ce produit scalaire.*

Définition 3.3.

Un noyau de Mercer : *est une fonction k telle que :*

$$\forall x, y \in \mathcal{X}, k(x, y) =< \Phi(x), \Phi(y) >$$

avec Φ une fonction qui, à un élément x de l'espace d'entrée $\mathcal{X}$, fait correspondre un élément $\Phi(x)$ dans l'espace Hilbertien H.

Définition 3.4.

La matrice de Gram *associée à la fonction noyau k est la matrice carrée G de taille $N \times N$ définie pour un ensemble de données $\{x_i\}_{i=1}^N$ tel que $G_{ij} = k(x_i, x_j)$.*

Toute fonction symétrique $k(x_1, x_2)$ de $\mathcal{L}_2(X^2)$ admet une décomposition de la forme :

$$k(x_1, x_2) = \sum_i \gamma_i \phi_i(x_1)\phi_i(x_2)$$

avec $\phi_i \in \mathcal{L}_2(X)$ et $\gamma_i \in \mathbb{R}$. Les éléments γ_i et ϕ_i intervenant dans cette expression correspondent aux fonctions propres et valeurs propres de l'opérateur intégral défini par le noyau k, soit :

$$\int k(x_1, x_2)\phi_i(x_1)dx_1 = \gamma_i \phi_i(x_2)$$

Une condition suffisante pour que $k(x_1, x_2)$ soit un produit scalaire est que les valeurs propres γ_i soient positives. Il en est ainsi, selon le théorème de Mercer, si et seulement si la condition suivante est satisfaite pour toute fonction f de $\mathcal{L}_2(X)$:

$$\int\int k(x_1, x_2)f(x_1)f(x_2)dx_1dx_2 \geq 0$$

Les fonctions k vérifiant cette relation sont appelées noyaux de Mercer.

Dans le cas discret, une condition équivalente à celle de Mercer est que toute matrice de Gram soit semi-définie positive.

Définition 3.5. *Un noyau $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est semi-défini positif sur $\mathcal{X}$*

si k est symétrique : $\forall (x, y) \in \mathcal{X}^2, k(x, y) = k(y, x)$

et vérifie la condition :

$\forall n \in \mathbb{N}(x_1, \ldots, x_n) \in \mathcal{X}^n$ et $\forall (c_1, \ldots, c_n) \in \mathbb{R}^n$, la matrice de Gram G_{ij} associée à k ($G_{ij} = k(x_i, x_j)$) est semi-définie positive :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij}c_ic_j \geq 0 \tag{3.1}$$

3.2.3 Noyaux non définis

Dans la section précédente, les noyaux étaient définis dans un espace de Hilbert où le produit scalaire est défini positif. Cependant, certains noyaux correspondent à un produit scalaire non défini. Par conséquent, nous ne pouvons plus utiliser les algorithmes basés sur les noyaux semi-définis positifs. Deux approches sont alors possibles pour traiter ce cas. La première approche consiste à vérifier que sur l'ensemble des données la matrice de Gram est définie positive. La deuxième approche est de travailler dans un autre espace de représentation : l'espace de Krein.

3.2.3.1 Matrice de Gram et ensemble de données

Afin de rester dans les méthodes basées noyaux semi-définis positifs, une approche est de travailler localement sur l'ensemble de données. Pour celà, il suffit que la matrice de Gram soit semi-définie positive sur la base.

Ainsi, nous devons vérifier que la matrice est bien définie positive sur cet ensemble. Dans ce cas, si toutes les valeurs propres de la matrice de Gram associée à la fonction symétrique k sont positives ou nulles, alors il existe une fonction de Mercer k' telle que $\forall x_i, x_j \in X^2,\ k(x_i, x_j) = k'(x_i, x_j)$.

Prenons par exemple le cas du noyau de Wallraven et al. [WCG03], qui s'appuie sur un calcul de maximum :

$$K(B_i, B_j) \quad = \quad \frac{1}{|B_i|} \sum_r \max_s k(b_{ri}, b_{sj}) + \frac{1}{|B_j|} \sum_s \max_r k(b_{ri}, b_{sj}) \tag{3.2}$$

avec $B_i = \{b_{ri}\}_r$ des ensembles d'éléments.

Outre sa pertinence pour la mise en correspondance, ce noyau est intéressant car l'utilisation d'un calcul de maximum permet d'utiliser des algorithmes rapides. Contrairement à ce qui est annoncé par leurs auteurs, cette fonction n'est pas de Mercer. Un contre-exemple est exhibé dans [Lyu05]. Cependant nous pouvons utiliser les algorithmes basés noyaux définis positifs sur l'ensemble des données sous réserve que la matrice de Gram soit positive sur cet ensemble.

Dans le cas des expérimentations menées dans cette thèse, nous avons constaté de façon systématique que les valeurs propres des matrices de Gram sont positives ou nulles ce qui nous a permis d'appliquer sur nos bases les algorithmes basés noyaux semi-définis positifs.

3.2.3.2 Espace de Krein

Si la matrice de Gram possède des valeurs propres de signes opposés, on peut alors se tourner vers des méthodes conçues pour les noyaux non-définis, par exemple les SVM de Haasdonk [Haa05] ou le Discriminant de Fisher (Haasdonk et Pekalska [HP08]). Dans ce cas le noyau non défini positif associé à la matrice G ne peut alors pas être plongé dans l'espace de Hilbert (Schölkopf et Smola [SS02]), mais dans un nouvel espace dit espace de Kreĭn qui permet de traiter ce genre de noyaux (Bognar [Bog74], Pekalska et Haasdonk [PH09] et Ong *et al.* [OMCS04]).

L'espace de Kreĭn se réfère à un espace vectoriel $\mathcal{K}$ muni d'un produit scalaire $< ., . >_{\mathcal{K}}$: $\mathcal{K} \times \mathcal{K} \rightarrow \mathbb{R}$ tel qu'il existe dans $\mathcal{K}$ une décomposition orthogonale $\mathcal{K} = \mathcal{K}_+ \oplus \mathcal{K}_-$, où $(\mathcal{K}_+, \kappa_+(.,.))$ et $(\mathcal{K}_-, \kappa_-(.,.))$ sont deux espaces de Hilbert séparables associés à des produits scalaires définis positifs. Le produit scalaire de $\mathcal{K}$ est la différence de κ_+ et κ_-, c'est à dire quels que soient $\xi_+, \xi'_+ \in \mathcal{K}_+$ et $\xi_-, \xi'_- \in \mathcal{K}_-$, nous obtenons :

$$< \xi_+ + \xi_-, \xi'_+ + \xi'_- >_{\mathcal{K}} = \kappa_+(\xi_+, \xi'_+) - \kappa_-(\xi_-, \xi'_-)$$

Il existe alors une relation entre l'espace de Kreĭn et la matrice G non définie positive. G peut être représentée par la différence entre 2 matrices semi-définie positives :

$$G = G_+ - G_-$$

Il en résulte que la fonction noyau associée à G peut-être plongée dans un espace de Kreĭn. L'astuce du noyau est valable aussi dans l'espace de Kreĭn.

3.2.4 Noyaux usuels sur vecteurs

Le noyau linéaire :

$$k(x_i, x_j) = < x_i, x_j > .$$

Les noyaux gaussiens sont par exemple fréquemment utilisés :

$$k(x_i, x_j) = exp^{\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)}.$$

avec $\sigma \in \mathbb{R}^+$

Les noyaux gaussiens sont des noyaux de type radial, indiquant qu'ils dépendent de la distance $\|x_1 - x_2\|$ entre les observations.

Les noyaux polynômiaux sont de la forme :

$$k(x_i, x_j) = (< x_i, x_j > \ + c)^q.$$

avec $c \in \mathbb{R}^+$ et $q \in \mathbb{N}$

Le noyau triangulaire :

$$k(x_i, x_j) = 1 - \frac{1}{\sigma^2}||x_i - x_j||^2.$$

3.2.5 Combinaison de noyaux

Les noyaux usuels peuvent servir de base pour la définition de nouveaux noyaux. En effet, la définition d'un noyau peut s'avérer complexe notamment pour la vérification de sa validité. Pour construire de nouvelles fonctions noyaux, une méthodologie courante est de combiner des fonctions noyaux usuelles en utilisant les propriétés suivantes :

- $\forall \lambda > 0$, k est une fonction noyau alors λk est une fonction noyau.
- $\forall p \geq 1$, k est une fonction noyau alors k^p est une fonction noyau.
- k et k' sont des fonctions noyaux alors $k + k'$ est une fonction noyau.
- k et k' sont des fonctions noyaux kk' est une fonction noyau.

3.2.6 Noyaux sur sacs

Dans le but d'illustrer les techniques de construction des noyaux, prenons le cas des représentations par sacs d'attributs, où un document x_i est représenté par un ensemble non ordonné de vecteurs $B_i = \{b_{ri}\}_r$ (appelé sac). S'il existe une fonction noyau k sur les éléments b_{ri}, alors il est possible de construire une fonction noyau K sur sacs B_i en posant :

$$\Phi(B_i) = \sum_r \phi(b_{ri}) \tag{3.3}$$

avec ϕ la fonction d'injection correspondant à k et Φ celle correspondant à K.

Il s'en suit que :

$$\begin{aligned} K(B_i, B_j) &= \langle \Phi(B_i), \Phi(B_j) \rangle \\ &= \sum_r \sum_s \langle \phi(b_{ri}), \phi(b_{sj}) \rangle \\ &= \langle \sum_r \phi(b_{ri}), \sum_s \phi(b_{sj}) \rangle \\ &= \sum_r \sum_s k(b_{ri}, b_{sj}) \end{aligned} \tag{3.4}$$

La fonction K est une fonction noyau et son expression ne dépend que des valeurs de la fonction k. D'après le lemme 1 d'Haussler [Hau99], K est défini positif si et seulement si k est défini positif.

Notons que k est généralement appelée fonction noyau *mineure* et K fonction noyau *majeure*.

3.2.7 Noyaux sur graphes

La définition d'un noyau calculé uniquement à partir des informations contenues dans les étiquettes des sommets et des arcs est possible, mais ne prend pas en compte l'organisation du graphe. Or cette structure de graphe apporte une information supplémentaire importante en définissant les relations entre chaque élément du graphe. Notre objectif consiste à tenir compte de cette information structurelle. Récemment, plusieurs travaux ont proposé des noyaux constitués d'ensembles non-ordonnés (Kondor et Jebara [KJ03] et Wallraven et al.[WCG03]). L'idée proposée par Wallraven et al., consiste ainsi à définir un noyau mineur entre chaque élément des ensembles et de regrouper ensuite les résultats fournis par les noyaux mineurs en un noyau de plus haut niveau qui définit ainsi un produit scalaire entre les deux ensembles. Une méthode de comparaison de graphes basée sur la comparaison des étiquettes de sommets et d'arcs peut donc être assimilée à des sacs de sommets et d'arcs. En étendant cette idée, nous pouvons définir des sacs de chemins pour comparer deux graphes. En effet, un chemin issu d'un graphe permet de définir un sous-graphe et structure les étiquettes des sommets et arcs présents sur ce chemin.

Il existe deux principales approches pour définir les noyaux sur graphes, en fonction de la façon dont la représentation du graphe dans l'espace vectoriel est effectuée.

La première appoche est la représentation explicite. Dans ce cas, des attributs sont extraits du graphe (nombre de sommets, chemins, spectre, ...). Dans

le but de choisir ces attributs, des prototypes sont construits en utilisant des techniques comme les K-Means, l'Analyse en Composantes Principales (Wilson et al. [WH03]), les techniques "Multiple Instance Learning" comme Chen et al. [CW04], ou encore les forêts alératoire (Moosmann et al. [MNJ08]). A partir de ces prototypes, une représentation explicite consiste par exemple à calculer la distance à chaque prototype, puis à utiliser un noyau classique sur vecteurs ou histogrammes (Bunke et al. [BR07]).

L'approche explicite limite la dimension des vecteurs dans l'espace induit, étant donné qu'ils doivent être stockés en mémoire. De plus, cela requiert des paramètres globaux (comme le nombre de prototypes) qui doivent être réglés pour chaque base de données ou pour chaque requête. Une solution à ce problème est de réaliser un calcul en ligne des paramètres et prototypes pendant la requête (Gondra et Heisterkamp [GH04] et Mairal et al. [MBPS09]).

La deuxième approche est la représentation implicite, qui permet de manipuler des représentations de très grandes dimensions sans jamais avoir à les calculer. De telles fonctions noyaux ont été proposées pour traiter les graphes sans arêtes (Eichhorn [EC04], Lyu [Lyu05] et Gosselin [GCPF07a]). Pour les graphes avec arêtes, certaines méthodes que nous avons citées précédemment ont été adaptées au contexte de la recherche multimedia et classification des formes, comme Suard et al. [SGRB05] et Dupé et Brun. [DB09]. Des techniques spectrales pour la mise en correspondance de paires de sommets ont aussi été proposées [LH05], ainsi que des méthodes avec les tenseurs pour les mises en correspondance d'ordres supérieurs (Bach [DBKP09]).

Les premiers noyaux considérés comme des noyaux sur graphes sont les noyaux de convolution (Haussler [Hau99]).

Considérons un espace $\mathcal{X}$ de données structurées (arbres, graphes, séquences...). Supposons qu'une donnée $x \in \mathcal{X}$ est décomposable en un ensemble de D sous-structures $x_1, ..., x_D$ où chaque x_i appartient à un espace $\mathcal{X}_i$. Nous pouvons alors définir une relation R sur $(\mathcal{X}_1, ..., \mathcal{X}_D, \mathcal{X})$ telle que $R(x_1, ..., x_D, x)$ soit vraie si le vecteur $\overrightarrow{x} = (x_1, ..., x_D)$ correspond effectivement aux sous-parties. Notons R^{-1} la relation inverse associée à la relation R.

Supposons maintenant que pour chaque espace $\mathcal{X}_i$ nous disposons d'un noyau K_i étant la similarité entre les données de cet espace. On peut alors définir un noyau de convolution pour $x, y \in \mathcal{X}$:

$$K(x,y) = \sum_{\substack{\overrightarrow{x} \in R^{-1}(x) \\ \overrightarrow{y} \in R^{-1}(y)}} \prod_{i=1}^{D} K_i(x_i, y_i).$$

Sous l'hypothèse que les fonctions K_i sont des noyaux, il est démontré que K est un noyau. Ceci est notamment dû aux propriétés de fermeture de la famille des noyaux (en particulier par les opérations de somme et de produit).

L'avantage de ces noyaux de convolution est qu'ils sont très généraux et peuvent s'appliquer à un grand nombre de problèmes (séquences, arbres, graphes...). Néanmoins, ils nécessitent une importante étape de mise au point pour choisir une relation R consistante avec le problème considéré ce qui ne facilite pas leur mise en oeuvre. De plus il faut également définir les différents noyaux K_i en s'assurant qu'ils vérifient les conditions générales des noyaux, et notamment le caractère semi-défini positif.

Ce modèle très généraliste est repris par Gärtner [Gär03]. D'autres noyaux ont aussi été proposés par la suite, et peuvent être répartis par familles en fonction du type de structure qu'ils considèrent.

3.2.7.1 Noyaux sur chemins

L'une de ces familles de noyaux est basée sur des chemins aléatoires. L'idée est de sommer les similarités entre les différents chemins des deux graphes. En pratique l'intégralité des chemins n'est pas considérée, et diverses approches sont proposées pour choisir le meilleur tirage. Par exemple, nous pouvons considérer que les chemins sont issus de marches aléatoire (Kashima et Tsuboi [KT04], Suard et al. [SGRB05]), ou encore s'appuyer sur le produit direct entre graphes (Borgwardt et al. [BOS+05] et Vishwanathan et al. [VSKB09]).

Une autre famille compare les graphes en sommant les similarités entre les sous-graphes élémentaires (*graphlets*) des deux graphes (Shervashidze et al. [SVP+09]). Ces méthodes sont motivées par les limites des chemins aléatoires, qui ne peuvent discerner certains graphes (Mahé et Vert [MV09]). En effet, deux graphes différents peuvent avoir les même ensembles de motifs (formés par des chemins aléatoires dans les graphes). Notons que ces cas arrivent surtout lorsque les graphes ont des sommets dont les similarités ont des valeurs binaires. Des propositions ont aussi été faites en considérant des arbres élémentaires [MV09, SB09].

Beaucoup de noyaux de la littérature ont été construits pour la chimie ou les applications bio-informatiques, où les sommets et les arêtes ont peu d'information, généralement une étiquette, voire un vecteur de faible dimension (moins de 4). De plus, ces méthodes ont été proposées pour des graphes de petite taille, mis à part certaines comme Shervashidze et al. [SVP+09], mais qui ne considèrent

que des sommets non étiquetés. Dans le contexte de la recherche multimedia, ces méthodes ont besoin d'être adaptées.

Dans le cadre de cette thèse, nous nous focalisons sur la dernière représentation, et plus particulièrement celle basée sur les chemins aléatoires. Ce type de fonction noyau nous semble être le plus adapté pour comparer les graphes dont les sommets et les arêtes portent une information riche.

Un chemin h dans un graphe $G = (V, E)$ est une séquence de sommets de l'ensemble V reliés par des arêtes appartenant à l'ensemble $E : h = (v_0, v_1, ..., v_n)$, $v_i \in V$. Nous considérons $H(.)$ une fonction permettant d'extraire d'un graphe un ensemble de chemins.

Dans les premiers papiers sur les noyaux de chemins, Kashima et Tsuboi [KT04] et Gärtner et al. [GFW03] utilisent des chemins aléatoires pour modéliser et comparer les graphes.

Kashima propose d'évaluer la similarité de deux graphes G et G' à l'aide d'un noyau comparant tous les chemins possibles de même longueur entre les deux graphes. Le noyau $K_{kashima}(G, G')$ est la somme pondérée des comparaisons entre tous les chemins h de G et h' de G' de même longueur :

$$K_{Kashima}(G, G') = \sum_{h \in H(G)} \sum_{\substack{h' \in H(G') \\ |h'|=|h|}} K_C(h, h') p(h|G) p(h'|G') \tag{3.5}$$

avec :
- $p(h|G)$ la probabilité de trouver le chemin h dans le graphe G
- $|h|$ la longueur du chemin h
- $K_C(h, h')$ fonction noyau qui compare deux chemins
- $H(G)$ ensemble des chemins possibles dans G

La fonction noyau $K_C(h, h')$ proposée par Kashima et al. [KT04] compare deux chemins de même longueur, et est basée sur un noyau sur sommets $K_V(v, v')$ et sur un noyau sur arêtes $K_E(e, e')$.

$$K_C(h, h') = \begin{cases} K_V(v_1, v'_1) \times \prod_{i=2}^{n} K_V(v_i, v'_i) K_E(e_i, e'_i) & \text{si } h = h' \\ 0 & \text{sinon} \end{cases}$$

Ce type de noyau est utilisé dans le cadre de graphes de molécules, où les sommets sont étiquetés par des symboles et donc la similitude entre les sommets est binaire, un sommet (un atome) est ou n'est pas le même que le sommet du graphe comparé. Mais quand les sommets sont valués par des valeurs réelles, un très grand nombre de petites valeurs de similarité entre chemins peuvent être ainsi

sommées, et finissent par diluer les similarités les plus fortes. Ces noyaux définis pour les graphes étiquetés (graphes de molécules) ont été étendus aux valeurs continues par Borgwardt [BOS+05]. Dans le papier de Vishwanathan [VSKB09], le noyau est calculé en comptant le nombre de marches aléatoires communes. Ces différents papiers calculent le noyau sur les graphes de toutes les marches aléatoires sur les graphes. Le nombre de ces marches peut être infini (en particulier si les cycles sont autorisés). Cependant, l'expression des probabilités $p(h|G)$ est récursive et permet d'exprimer le problème sous la forme d'un système fini. Les valeurs des probabilités et celles des noyaux sur sommets et arêtes étant inférieures à 1, la convergence du calcul est assurée.

3.2.7.2 Noyaux non définis

Le problème des noyaux précédents est la haute complexité de calcul. Si celà est acceptable avec des graphes de molécules chimiques, qui ont des valeurs symboliques, ce n'est pas le cas avec nos graphes attribués. Le noyau K_{max} prend le maximum de toutes les similitudes de tous les chemins de même longueur.

$$K_{max}(G, G') = \max_{h \in H(G)} \max_{\substack{h' \in H(G') \\ |h'|=|h|}} K_C(h, h') \tag{3.6}$$

Un noyau similaire fut utilisé dans FReBIR (Philipp et al. [PFG06]) sans la restriction sur la longueur des chemins. Suard [SGRB05, Sua06] propose une version approximée de la méthode de Kashima en s'inspirant du noyau K_{max}. Son principal objectif est de réduire la complexité calculatoire. Il propose de moyenner sur les meilleurs appariements plutôt que de faire la moyenne sur tous les appariements possibles comme Kashima. Le noyau proposé est donc le suivant :

$$K_{Suard}(G, G') = \frac{1}{2} \left(\sum_{h \in H(G)} \max_{\substack{h' \in H(G') \\ |h'|=|h|}} K_C(h, h') + \sum_{h' \in H(G')} \max_{\substack{h \in H(G) \\ |h|=|h'|}} K_C(h, h') \right) \tag{3.7}$$

avec $H(G)$ une fonction qui génère un ensemble de chemins issus du graphe G.

La méthode de Suard propose aussi de considérer un ensemble de chemins moins vaste $H(G)$, au lieu de considérer tous les chemins possibles. Il considère l'ensemble des plus courts chemins entre deux sommets du graphe. Il n'y a pas de chemin avec boucle ou cycle. Sur la figure (Fig.(3.2)) et le tableau (Tab.(3.1)),

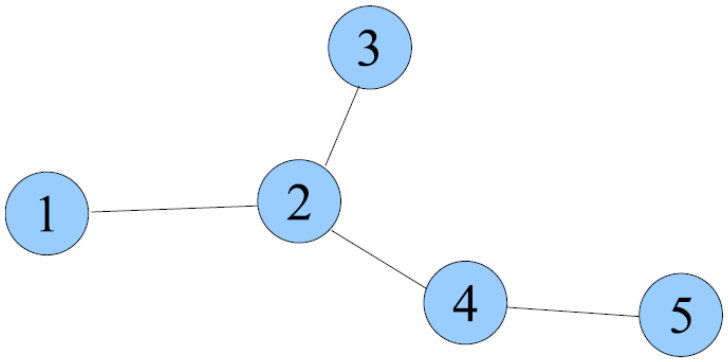


FIGURE 3.2 – Graphe exemple

Longueur	$H(G)$ chemins les plus courts	chemins aléatoires
0	1	1
1	12	12
2	123 , 124	121 , 123 , 124
3	1245	1212 , 1232 , 1243 , 1245 , 1242

TABLE 3.1 – Exemple des chemins issus du graphe 3.2 qui commencent par le sommet 1.

nous montrons un exemple de la sélection des plus courts chemins pour les chemins débutant par le sommet 1.

Ainsi, il y a au plus $|V|^2$ (resp. $|V'|$) chemins considérés dans G (resp. G'), et donc au plus $|V|^2 \times |V'|^2)$ comparaisons de chemins.

Un autre noyau sur graphe fut proposé par Lebrun et al. [LPFG08]. L'auteur propose de faire une moyenne des meilleures similarités d'appariements de chemins. De plus, l'ensemble des chemins à comparer est réduit. Afin d'être certain de considérer chaque sommet au moins une fois, les auteurs considèrent un chemin h_{v_i} pour chaque sommet v_i. La fonction génératrice $H_{v_i}(G)$ fournit l'ensemble des chemins qui commencent par le sommet v_i. De plus, lors de la recherche du meilleur appariement, le nombre de chemins dans G' est réduit, en ne gardant, dans l'ensemble de recherche que les chemins qui commencent par le sommet $v'_i \in V', v'_i = s(v_i)$ le plus similaire à v_i. Nous avons $H_{s(v'_i)}(G)$ qui génère

l'ensemble des chemins qui débutent par le sommet le plus similaire à v_i.

$$\begin{aligned} K_{Lebrun}(G,G') &= \frac{1}{|V|}\sum_{i=1}^{|V|} \max_{\substack{h_{v_i}\in H_{v_i}(G) \\ h'_{s(v_i)}\in H_{s(v_i)}(G') \\ |h_{v_i}|=|h'_{s(v_i)}|}} K_C(h_{v_i}, h'_{s(v_i)}) \\ &+ \frac{1}{|V'|}\sum_{i=1}^{|V'|} \max_{\substack{h_{s(v'_i)}\in H_{s(v'_i)}(G) \\ h'_{v'_i}\in H_{v'_i}(G') \\ |h_{s(v'_i)}|=|h'_{v'_i}|}} K_C(h_{s(v'_i)}, h'_{v'_i}). \end{aligned} \tag{3.8}$$

Dans son noyau, Lebrun cherche à combiner les chemins de l'ensemble généré par $H_{v_i}(G) = \{h | v_i$ est le premier sommet du chemin$\}$ avec les chemins de G' dont le premier sommet est $v'_i \in V$ tel que $v'_i = s(v_i)$ est le plus similaire à v_i. Cette propriété est intéressante pour les graphes de régions car les régions (définissant les sommets du graphe) portent énormément d'information, mais dans notre étude de graphes de segments de contours, l'information est plus portée par la structure du graphe que par les sommets eux-mêmes.

3.3 Séparateurs à Vaste Marge SVM

L'approche de la similarité par fonction noyau consiste à utiliser un produit scalaire comme fonction de similarité. D'une manière générale, cette approche permet de transformer un problème non linéaire en un problème linéaire plus simple à résoudre (convexe, stable) permettant aussi d'exploiter des méthodes de recherche de l'hyperplan séparateur optimal (Fig.(3.3)).

Le problème du classement binaire par un hyperplan consiste à trouver un hyperplan séparant l'espace des données en un sous-espace d'instances positives et un sous-espace d'instances négatives. Plus formellement, étant donné un ensemble de données $D = (\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_N}, y_N))$ d'apprentissage (les étiquettes y_i sont connues) avec $x \in \mathbb{R}^n$ et $y \in -1, +1$, le problème consiste à déterminer un hyperplan de vecteur normal $\boldsymbol{w}$ tel que :

$$\forall(\boldsymbol{x_i}, y_i) \in D : y_i(< \boldsymbol{w}, \boldsymbol{x_i} > + b) \geq 0$$

L'étiquette y pour le vecteur $\boldsymbol{x}$ non-encore étiqueté est alors déterminée par :

$$y = sign(< \boldsymbol{w}, \boldsymbol{x} > + b) \tag{3.9}$$

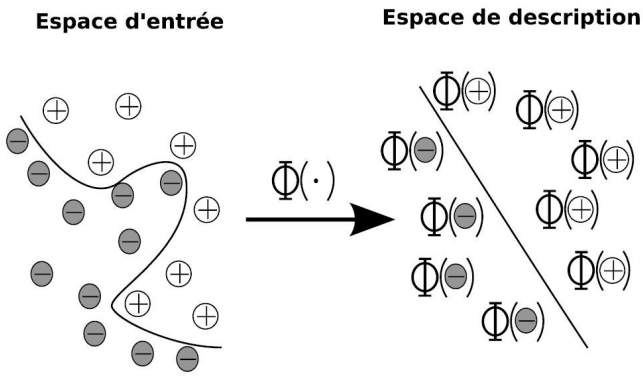


FIGURE 3.3 – Passage d'un espace non-linéairement séparable vers un espace de redescription, à l'aide d'une fonction d'injection Φ, linéairement séparable.

Les Séparateurs à Vaste Marge, ou machines à vecteur support (Support Vector Machines, SVM) sont des algorithmes de classification binaire issue de la théorie de Vapnik [Vap98]. Le principe des Séparateurs à Vaste Marge est de trouver l'hyperplan séparateur entre les deux classes constituant l'ensemble de données étudié et qui maximise la marge entre ces deux classes. Le principe d'hyperplan à marge maximale est illustré sur la figure (Fig.(3.4)) pour un espace à deux dimensions.

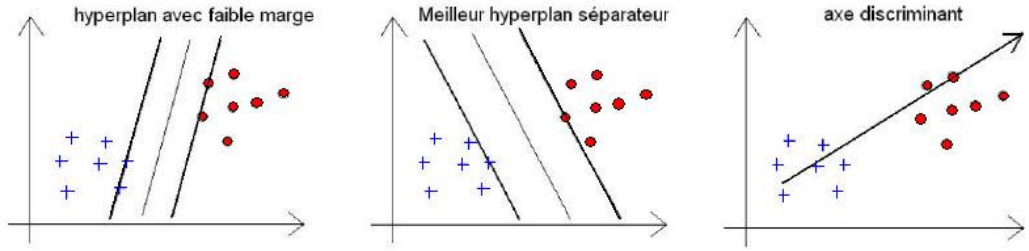


FIGURE 3.4 – Illustration de la recherche de l'hyperplan à marge maximale dans un espace à deux dimensions.

Parmi les algorithmes de SVM, on considère les cas linéairement séparables et les cas non linéairement séparables. Les premiers sont les plus simples car ils permettent de trouver facilement le classificateur linéaire. Dans la plupart des problèmes réels il n'y a pas de séparation linéaire possible entre les données, le classificateur de marge maximale ne peut donc pas être utilisé tel quel. Pour surmonter les inconvénients des cas non linéairement séparables, l'idée des SVM est de changer l'espace des données et permettre l'utilisation des fonctions

noyaux. Ces dernières s'intègrent parfaitement au cadre théorique des SVM puisqu'une fonction noyau K peut toujours s'écrire sous la forme d'un produit scalaire $K(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >$ même si Φ n'est pas explicite et elle permet donc aux SVM de définir l'hyperplan séparateur dans l'espace induit par Φ en résolvant :

$$sign(\sum_{i=1}^{N} \alpha_i.y_i.k(\boldsymbol{x_i}, \boldsymbol{x})).$$

Les coefficients α_i sont appris pour maximiser la classification sur les données d'apprentissage.

Chapitre 4

Méthodes proposées de calcul de similarité entre graphes

Dans le cadre de la mise en correspondance inexacte des graphes, nous proposons plusieurs variantes d'une nouvelle similarité basée sur des ensembles de chemins tracés sur les graphes, capables d'effectuer les groupements des contours et robustes aux changements d'échelle. D'après la définition 2.7, un chemin est une suite de sommets reliés les uns aux autres par des arcs (sommets et arêtes sont distincts). Dans notre cas, il s'agit donc d'une suite de contours voisins. Sur le graphe (Fig.(4.1)), nous avons extrait du graphe de fenêtre un chemin possible. Afin de comparer les différents chemins, nous proposons une similarité entre chemins qui prend en compte la similarité des ensembles de segments de contours. Puis nous proposons une combinaison de similarité sur contours associée à une similarité des régions définies par ces chemins.

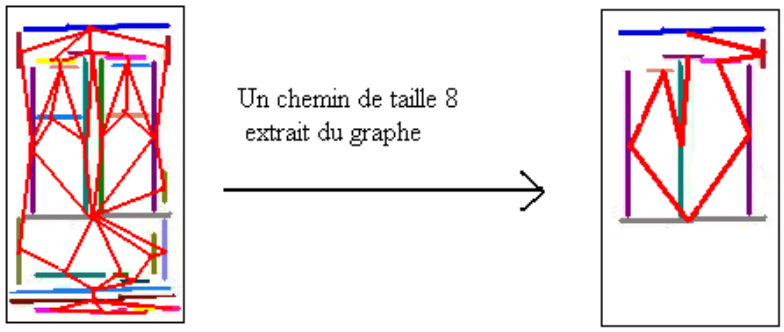


FIGURE 4.1 – Les chemins du graphe de contours sont une suite de contours proches. Sur le graphe ci-dessus, un chemin de taille 8 est extrait.

4.1 Noyau sur graphes proposé

4.1.1 Noyau sur graphes, noyaux sur chemins

Nous proposons de reprendre le noyau de Lebrun *et al.* mais en supprimant la contrainte sur le sommet de départ. En effet dans notre cas, si nous nous restreignons dans G' à l'ensemble des chemins dont le sommet de départ est le plus similaire au sommet v_i de G, nous obtenons souvent de mauvais appariements. Etant donné que les contours pris séparément sont peu discriminants (beaucoup de contours dans la même orientation), il existe alors de nombreux bons appariements possibles entre un contour de G et un contour de G' et se restreindre à un parmi tant d'autres entraîne une mauvaise sélection de chemins possibles dans G' avant même la recherche du meilleur appariement entre chemins.

Notre similarité est donc basée sur le noyau de Wallraven [WCG03], Suard [Sua06] mais la fonction génératrice de chemins et les noyaux mineurs diffèrent :

$$\begin{aligned} K_{struct}(G,G') &= \frac{1}{|V|}\sum_{i=1}^{|V|} \max_{\substack{h'\in H(G') \\ h_{v_i}\in H_{v_i}(G) \\ |h'|=|h_{v_i}|}} K_C(h_{v_i},h') \\ &+ \frac{1}{|V'|}\sum_{i=1}^{|V'|} \max_{\substack{h\in H(G) \\ h'_{v_i}\in H_{v'_i}(G') \\ |h|=|h'_{v'_i}|}} K_C(h,h'_{v'_i}). \end{aligned} \tag{4.1}$$

avec $H_{v_i}(G)$ la fonction génératrice de chemins qui au graphe G fait correspondre un ensemble de chemins issus de G et qui commencent par le sommet v_i. $H(G)$ la fonction génératrice de chemins qui au graphe G fait correspondre un ensemble de chemins issus de G. Ces chemins ne contiennent pas de boucles et de cycles.

Cette fonction est symétrique afin de la considérer comme un noyau, malgré le fait que les conditions de Mercer ne soient pas vérifiées dans certains cas mais elles s'avèrent toujours vérifiées sur les bases de données que nous avons utilisées.

Concernant les noyaux sur chemins K_C, plusieurs noyaux furent proposés par Kashima [KT04] et Lebrun [LPFG08] (somme, produit,...).

Le premier noyau K_C considéré effectue le produit entre toutes les similarités des sommets et arêtes composant les 2 chemins :

$$K_{C_{mul}}(h,h') = \begin{cases} K_v(v_0,v'_0) \ \times \prod_{j=1}^{|h|} K_e(e_j,e'_j)\ K_v(v_j,v'_j) & \text{si } |h| = |h'| \\ 0 & \textit{sinon} \end{cases} \quad (4.2)$$

où e_j est l'arc entre (v_{j-1} et v_j) :

K_v et K_e sont les noyaux mineurs qui définissent la similarité entre les sommets et la similarité entre les arcs.

Ce noyau pénalise cependant les longs chemins. En effet, si la similarité d'un sommet ou d'une arête est mauvaise ($K_e < 1$ ou $K_v < 1$), alors la similarité noyau sur les chemins passant par ce sommet ou cette arête décroit. Ainsi lors de la recherche des meilleurs chemins, les courts chemins sont favorisés.

Le deuxième noyau effectue la somme entre toutes les similarités des sommets et arêtes composant les 2 chemins :

$$K_{C_{som}}(h,h') = \begin{cases} K_v(v_0,v'_0) \ + \sum_{j=1}^{|h|} K_e(e_j,e'_j)\ K_v(v_j,v'_j) & \text{si } |h| = |h'| \\ 0 & \textit{sinon} \end{cases} \quad (4.3)$$

La similarité noyau augmente avec la taille des chemins si les noyaux mineurs sont positifs. Ainsi les chemins courts ont une plus faible similarité que les chemins prolongés.

Le troisième noyau permet une augmentation de la similarité avec la longueur des chemins. Cependant, en utilisant un produit, les auteurs pénalisent fortement tout couple de chemin qui ont au moins un couple sommet/arête non similaire :

$$K_{C_{mul1}}(h,h') = \begin{cases} K_v(v_0,v'_0) \ \times \prod_{j=1}^{|h|} (1 \ + \ K_e(e_j,e'_j)\ K_v(v_j,v'_j)) & \text{si } |h| = |h'| \\ 0 & \textit{sinon} \end{cases} \quad (4.4)$$

Le dernier noyau traite différemment les similarités des sommets et les similarités des arêtes composant les 2 chemins :

$$K_{C_{mul2}}(h,h') = \begin{cases} K_v(v_0,v'_0) \ \times \prod_{j=1}^{|h|} (K_e(e_j,e'_j) \ \times (1 \ + \ K_v(v_j,v'_j))) & \text{si } |h| = |h'| \\ 0 & \textit{sinon} \end{cases} \quad (4.5)$$

La similarité $K_{C_{mul2}}$ donne un poids plus important aux similarités d'arêtes (les relations spatiales).

Nous avons testé ces noyaux et les meilleurs résultats sont obtenus avec le noyau $K_{C_{som}}$ (eq.(4.3)). L'avantage de ce noyau est qu'il favorise les longs chemins et donc décrit mieux la structure de l'objet dans notre cas. L'ajout d'une arête et d'un sommet apporte une information en plus dans ce noyau. Nous devons maintenant définir les noyaux mineurs qui vont permettre de mesurer les similarités entre sommets et entre arêtes, et donc caractériser les contours et leurs relations.

4.1.2 Noyau sur chemins proposé

Nous avons vu dans la partie sur les attributs de sommets du chapitre 2 que nous souhaitions comparer l'orientation des contours. Ainsi un sommet v_i est caractérisé par le vecteur $(cos(2\Theta_i), sin(2\Theta_i))^T$ avec Θ_i l'angle formé entre le contour C_i et l'axe horizontal. Sur le tableau (Tab.(2.1)) du chapitre 2, nous montrons les différents produits scalaires entre les représentations vectorielles des contours horizontaux et verticaux. Nous constatons que le produit scalaire est le plus faible pour des contours orthogonaux et le plus élevé pour des contours parallèles. Nous proposons d'utiliser le produit scalaire comme mesure de similarité pour le noyau mineur sur les sommets. Nous proposons d'utiliser le noyau K_v basé sur le noyau cosinus :

$$K_v(v_j, v'_j) = \frac{\langle v_j, v'_j \rangle}{||v_j||.||v'_j||} + 1.$$

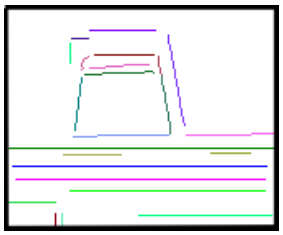
Notons que nous calculons le cosinus de l'angle formé entre le vecteur v_j et v'_j. Dans notre cas, v_j et v'_j sont unitaires. De plus nous ajoutons $+1$ dans la similarité afin d'avoir des similarités positives et de situer la similarité entre $[0, 2]$ et non entre $[-1, 1]$.

Le noyau sur arêtes K_e est basé sur le même principe. L'arête dans le graphe est caractérisée par la position relative des centres de gravité des segments de contours C_i and C_j, notés $g_{C_i}(Xg_{C_i}, Yg_{C_i})$ et $g_{C_j}(Xg_{C_j}, Yg_{C_j})$. Une arête est désignée par le vecteur $e_{ij} = (Xg_{C_j} - Xg_{C_i}, Yg_{C_j} - Yg_{C_i})^T$. Nous souhaitons une forte similarité pour les vecteurs colinéaires et orientés dans le même sens et une faible similarité pour les vecteurs colinéaires mais opposés. C'est donc l'angle entre les vecteurs qui nous intéresse, et plus particulièrement la valeur du cosinus de l'angle entre les vecteurs. Nous proposons donc comme noyau mineur d'arêtes :

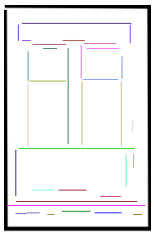
$$K_e(e_j, e'_j) = \frac{\langle e_j, e'_j \rangle}{||e_j||.||e'_j||} + 1.$$

Notre noyau vise à comparer des ensembles de contours, du point de vue de leur orientation et de leur position relative. Toutefois, quelques chemins peuvent avoir une forte valeur de similarité, mais ne fournissent aucune information structurelle, par exemple, des chemins dont tous les sommets représentent les segments de contours presque parallèles (Fig.(4.2)).

(a) Contours d'une fenêtre, sommets du graphe G.



(b) Contours d'un toit et cheminée, sommets du graphe G'.



(c) Contours d'une autre fenêtre, sommets du graphe G''.

FIGURE 4.2 – Exemple : trouver les meilleurs appariements de chemins de contours. Sans notre pondération le graphe G est plus proche de G' que de G''. Le problème est dû à la forte présence de chemins ne possédant que des contours horizontaux dans G et G'. Cependant ces chemins n'apportent aucune information sur la structure de l'objet.

Pour faire face à ce problème, nous pouvons augmenter la longueur des chemins, mais la complexité de calcul devient rapidement prohibitive. Pour surmonter ce problème, nous proposons d'ajouter à K_C un poids $O_{i,j}$ qui pénalise les chemins dont les orientations des différents segments consécutifs ne varient pas.

$$O_{i,j} = sin(\phi_{ij}) \times sin(\phi'_{ij}) = \sqrt{\tfrac{1}{2}(1 - \langle v_i, v_j \rangle)} \times \sqrt{\tfrac{1}{2}(1 - \langle v'_i, v'_j \rangle)}.$$

avec ϕ_{ij} (respectivement ϕ'_{ij}) l'angle entre les sommets i et j dans G (G').

De plus, le groupement perceptuel de l'ensemble de contours est essentiel pour la reconnaissance. Par exemple dans la figure (Fig.(4.3)) les graphes G' et G'' ont presque la même structure que le graphe G, mais le contour le plus à droite est plus loin dans le graphe G' que dans les deux autres graphes.

La question est : est-ce que ce contour doit être regroupé avec les autres pour former la structure d'un objet ou non ? Pour modéliser cette information, nous ajoutons un facteur d'échelle $S_{e_i e'_i}$.

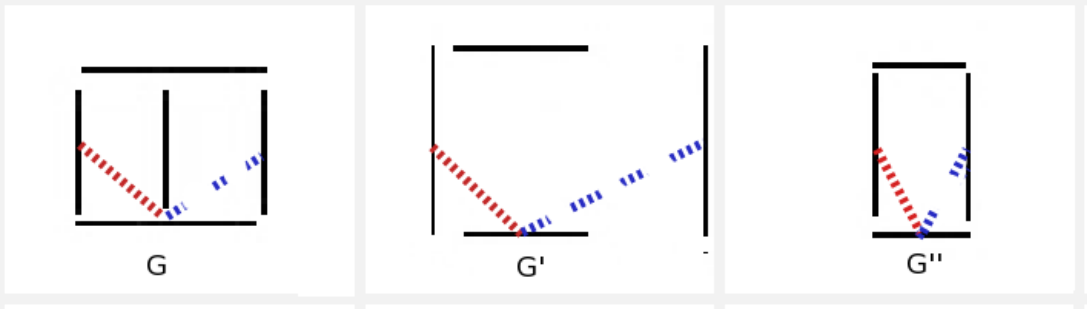


FIGURE 4.3 – Exemple : structure et problème d'échelle. Le segment de contour sur la droite du graphe G' est-t-il un contour de l'objet cherché ou non ?

Le facteur d'échelle $S_{e_i e'_i}$ compare le rapport d'échelle entre des appariements successifs d'arêtes. Pour un appariement entre l'arête $e_{i-1} \in E$ et $e'_{i-1} \in E'$ avec $G = (V, E)$ et $G' = (V', E')$, si la norme de e'_{i-1} est le double de e_{i-1}, nous voudrions garder le même facteur d'échelle lors de l'appariement suivant le long du chemin entre $e_i \in E$ et $e'_i \in E'$. La norme de e'_i doit être alors le double de celle de e_i. Ce poids $Se_i e'_i$ permet de conserver la structure de l'objet et favorise les appariements d'arêtes qui ont des facteurs d'échelle quasi-égaux. Nous proposons donc le poids suivant :

$$S_{e_i e'_i} = \min_{\substack{e_i \in h \\ e'_i \in h'}} \left(\frac{\frac{||e_{i-1}||}{||e'_{i-1}||}}{\frac{||e_i||}{||e'_i||}} \right)$$

Plus le rapport d'échelle $S_{e_i e'_i}$ est proche de 1, plus l'appariement est bon. Dans le cas $S_{e_i e'_i} > 1$, nous pénalisons l'appariement en prenant $S_{e_i e'_i} = \frac{1}{S_{e_i e'_i}}$. Nous avons donc le poids $S_{e_i e'_i} \in [0, 1]$.

Nous avons testé sur une base "jouet" de quelques graphes (Fig.(4.4)). L'objectif est de voir si les contours éloignés sont encore considérés comme contours de l'objet. Nous remarquons que le graphe avec un contour à droite éloigné des autres contours est alors pénalisé, ne sachant pas si ce contour fait réellement partie de l'objet rectangulaire recherché.

(a) Classement de la base "jouet" sans le poids échelle

(b) Classement de la base "jouet" avec le poids échelle

FIGURE 4.4 – Test sur une base "jouet" du poids échelle proposé : le graphe requête est l'imagette en haut à gauche. Les autres images de gauche à droite et de haut en bas sont classées par ordre de similarité. La structure du graphe requête est mieux évaluée avec la pondération échelle.

Nous obtenons au final :

$$S_{e_i e'_i} = \min_{\substack{e_i \in h \\ e'_i \in h'}} \left(\frac{||e_i||}{||e_{i-1}||} \cdot \frac{||e'_{i-1}||}{||e'_i||}, \frac{||e_{i-1}||}{||e_i||} \cdot \frac{||e'_i||}{||e'_{i-1}||} \right).$$

Notre noyau final K_C devient ($S_{e_i e'_i} \in [0,1]$ et $O_{i,j} \in [0,1]$) :

$$K_C(h_{v_i}, h') = K_v(v_i, v'_0) \; + \sum_{j=1}^{|h|} S_{e_i e'_i} O_{j,j-1} K_e(e_j, e'_j) \; K_v(v_j, v'_j). \tag{4.6}$$

4.1.3 Application : classification des hypothèses de fenêtres

Dans le chapitre 2, nous avons vu que l'algorithme proposé permet de trouver des hypothèses de fenêtres. Le problème reste que nous détections encore beaucoup de fausses-alarmes. L'utilisation de notre noyau (eq.(4.1) et eq(4.6)) associé à un classifieur Séparateurs à Vaste Marge pourrait différencier les fenêtres des faux-positifs. Nous rappelons que malgré le fait que pour le noyau proposé les conditions de Mercer ne soient pas vérifiées dans certains cas, elles s'avèrent toujours vérifiées sur les bases de données que nous avons utilisées.

Un exemple est montré ci-dessous, mais des évaluations plus complètes et le détail de la base seront présentés dans la partie résultat (chapitre 5). Nous avons extrait des hypothèses de fenêtres et formé une base de 220 images avec 70 fenêtres et 150 fausses alarmes (*cf.* chapitre 5, Fig.(5.4)).

Un utilisateur peut rechercher une catégorie d'images à l'aide de l'interface graphique RETIN [GCPF07b], dont une saisie d'écran est présentée en figures (Fig.(4.5), Fig.(4.6), Fig.(4.7), Fig.(4.8)). L'interface se décompose en deux parties principales. La première, qui prend la majeure partie de l'écran, présente une partie du classement de la base par le système. Par exemple, sur la figure (Fig.(4.5)) nous pouvons voir les images les plus pertinentes selon le système. Le classement est fait de gauche à droite puis de haut en bas. L'image de fenêtre avec un petit carré vert est l'image requête, puis celle à sa droite est la première plus pertinente, etc. La partie inférieure présente les images sélectionnées par la technique d'apprentissage actif. Sur la figure (Fig.(4.5)), le système a été configuré pour sélectionner 5 images. Ces 5 images sont déterminées par une stratégie active (Tong et Koller [TK01], Gosselin et Cord [GC06]) de façon à ce qu'ajouter les 5 étiquettes à l'ensemble d'apprentissage optimise le classifieur par rapport à l'itération de recherche précédente.

Nous présentons sur les figures (Fig.(4.5), Fig.(4.6), Fig.(4.7), Fig.(4.8)) les différentes étapes d'une session de recherche de la catégorie fenêtre. L'utilisateur a initialisée la requête en annotant une image de fenêtre. Le système a alors classé les images de la base en fonction de leur similarité à cette image requête. Sur la figure (Fig.(4.5)), nous pouvons voir ce classement : l'image en haut à gauche avec un petit carré vert est l'image annotée positivement, et les images qui suivent

sont ses plus proches voisines par rapport à la similarité. L'utilisateur, afin de raffiner sa requête, annote les images parmi les 5 images dans la partie inférieure de l'interface. L'utilisateur fournit ses annotations en cliquant sur les images. Les annotations sont représentées par des surimpressions de carrées vert (resp. rouge) pour les annotations positives (resp. négatives). Une fois les annotations données, l'utilisateur demande une mise à jour du classement (Fig.(4.6)). Puis nous itérons les annotations. Ces annotations se sont pas encore prises en compte, et sont affichées deux fois : une fois dans la barre de sélection, et une fois dans la partie principale (Fig.(4.7)). Les annotations négatives permettent d'éliminer des images qui ne sont pas des fenêtres de la tête du classement. Au bout de quelques annotations une grande partie des fenêtres se retrouvent en tête du classement (Fig.(4.8)).

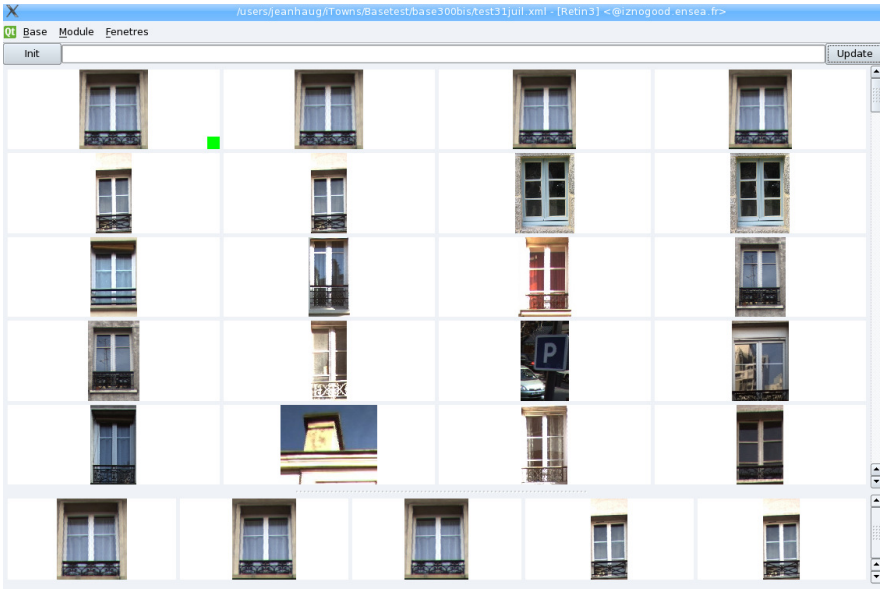


FIGURE 4.5 – Exemple d'une session de recherche sur une base de 220 imagettes avec 70 fenêtres. L'image en haut à gauche avec un petit carré vert est l'image annotée positivement, et les images qui suivent sont ses plus proches voisines. Le classement est fait de gauche à droite puis de haut en bas.

FIGURE 4.6 – Classement après une itération avec 3 labels positifs pour affiner la recherche.

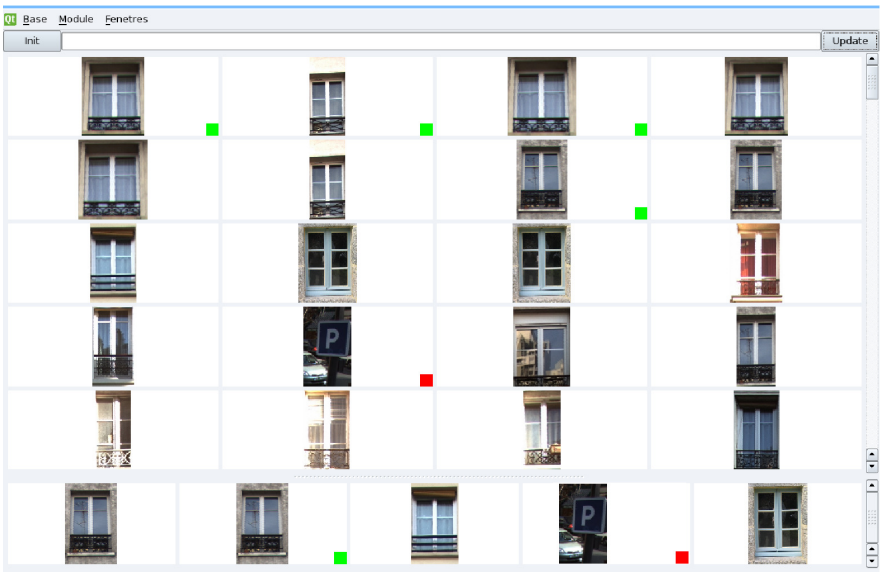


FIGURE 4.7 – Annotations suite au classement précédent. Nous annotons les éléments les plus pertinents sélectionnés par le système.

FIGURE 4.8 – Résultats après 4 itérations : 5 images annotées positivement et 4 négativement.

4.2 Recherche de sous-graphes similaires dans des graphes de contours

4.2.1 Extraction de fenêtres dans les façades

Dans le paragraphe précédent, nous avons introduit et utilisé les graphes globaux de contours pour classer des imagettes contenant des hypothèses de fenêtres selon leurs similarités. Précédemment, nous utilisions des noyaux dans le cadre des séparateurs à vaste marge afin de classer ces imagettes. Dans ce paragraphe, nous proposons des similarités de graphes non symétriques. N'étant plus symétriques, ce ne sont pas des noyaux, et nous avons donc pris comme classifieur un k-Plus-Proche-Voisin. De plus, dans ce paragraphe nous voulons nous affranchir de l'étape d'extraction d'hypothèses de fenêtres et exploiter directement le graphe des façades pour localiser la position des sous-graphes fenêtres. Nous cherchons donc à apparier les graphes requêtes avec des sous-graphes des façades. Ceci nous permettra non seulement de classer les façades en fonction de la fenêtre requête mais aussi de localiser précisément les fenêtres dans les façades. Le premier aspect répond à une requête du type "trouver des façades de type Haussmannien", à partir d'un exemple ou d'un ensemble d'exemples de fenêtres Haussmanniennes. La localisation des fenêtres a des applications pour la modélisation ou la reconstruction procédurale des bâtiments.

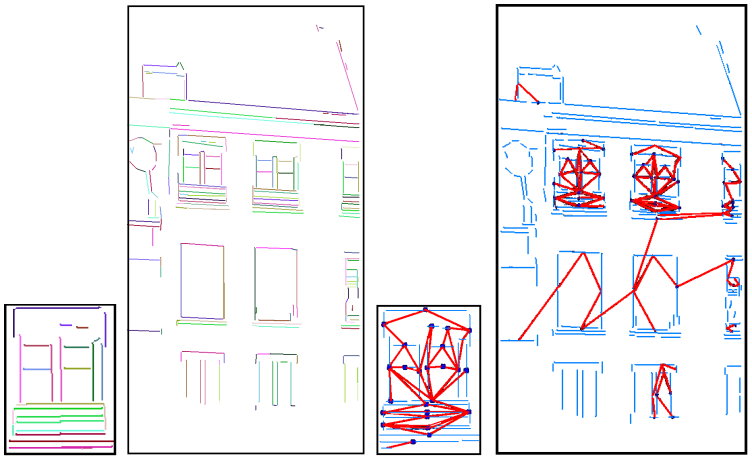
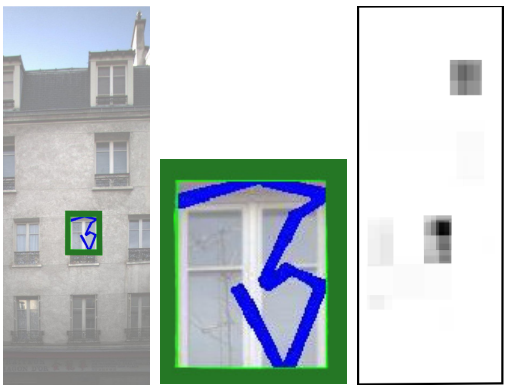


FIGURE 4.9 – **Les meilleurs appariements. (a)** Graphe requête. **(b)** Graphe façade (pour une meilleure lisibilité, les arêtes ne sont pas tracées). **(c)(d)**(traits rouges) Chemins appariés du sous-graphe requête (a) et du graphe (b). On constate que la majorité des chemins est localisée sur des régions pertinentes.

Nous utiliserons la notion de "chemin d'intérêt", qui est un chemin du graphe cible G' apparié avec un chemin du graphe requête G. Un chemin d'intérêt déterminera une "région de chemin d'intérêt" que nous définirons dans le paragraphe suivant. Nous pourrons alors introduire des similarités exploitant à la fois la similarité des contours et des régions d'intérêt définies par ces contours.

4.2.1.1 Région de chemin d'intérêt

Dans la similarité définie par l'equation (eq.(4.1)), pour chaque sommet v_i de G, nous cherchons le meilleur appariement entre le chemin $h_{v_i} \in G$ (chemin débutant par le sommet v_i et un chemin $h' \in G'$) (Fig.(4.9)).



(a) (b) (c)

FIGURE 4.10 – **Localisation des fenêtres. (a)** Un chemin et sa région d'intéret dans la façade. **(b)** Zoom sur le chemin. **(c)** Image J obtenue par accumulation des votes de régions d'intérêt.

A partir du meilleur h' retrouvé, nous définissons une Région de Chemin d'Intéret (RCI) autour de ce chemin h'. Nous définissons la RCI comme le rectangle dont les côtés sont parallèles aux bords de l'image et qui contient tous les centres de gravité des segments de contours appartenant au chemin. Afin de localiser les fenêtres dans le graphe des façades, nous affectons une valeur d'évaluation à chaque pixel de la RCI. Cette valeur peut être un simple vote (+1 à chaque pixel de la RCI) ou un vote pondéré (par la similarité de l'appariement). Tous les meilleurs chemins h' appariés à un chemin h_{v_i} de G vont donc voter pour un ensemble de pixels dans l'image représentée par G'. Le résultat (Fig.(4.10)) est une accumulation de votes de ces régions de chemins d'intérêt, ce qui s'écrit pour chaque pixel (x,y) de l'image :

$$J_{G,G'}(x,y) = \sum_{i=1}^{|V|} J_{h_{v_i},h'}(x,y) \tag{4.7}$$

où $J_{h_{v_i},h'}(x,y)$ vaut la valeur d'évaluation. Dans notre cas, nous avons choisi un vote pondéré $J_{h_{v_i},h'}(x,y) = S_C(h_{v_i},h^{'})$ où $S_C(h_{v_i},h^{'})$ est la valeur de similarité entre le chemin h_{v_i} et $h^{'}$.

$$S_C(h_{v_i},h') = K_v(v_i,v'_0) \ + \sum_{j=1}^{|h|} S_{e_i e'_i} O_{j,j-1} K_e(e_j,e'_j) \ K_v(v_j,v'_j). \tag{4.8}$$

De plus, nous souhaitons mesurer la similarité d'un graphe de fenêtre comme sous-graphe dans les graphes de façades. Or dans l'équation (eq.(4.1)), seul le premier terme nous intéresse :

$$\tfrac{1}{|V|} \sum_{i=1}^{|V|} \max_{h_{v_i},h^{'}} \ S_C(h_{v_i},h^{'}).$$

Ainsi, nous avons décidé de ne plus respecter la symétrie et de seulement mesurer la similarité dans un sens (trouver un graphe de fenêtre dans un graphe de façade). N'étant plus symétrique, nous avons pris comme classifieur un k-Plus Proche Voisin.

Dans la section précédente, nous avons mis en évidence le besoin d'avoir de nombreux chemins pour décrire l'objet et d'atténuer les chemins de bruit. Pour celà, nous choisissions plusieurs chemins de taille fixe. Notons que selon l'application, la longueur des chemins peut être très variable. Kashima [KT04] somme sur toutes les longueurs de chemins tracés sur des molécules chimiques. Lebrun [LPFG08] a montré que des chemins de longueur 1 étaient suffisants pour des graphes de régions. Dans le premier cas, l'information portée par les sommets est très réduite (un symbole chimique), dans le deuxième cas, elle est très riche (couleur et texture). Dans notre cas, nous proposons maintenant de rechercher les meilleurs chemins débutant par chaque sommet mais dont la longueur varie. L'algorithme de "séparation et évaluation" (*cf.* chapitre 5) permet d'explorer les branches de l'arbre de recherche les plus intéressantes en terme de similarité. Ainsi si l'arbre est exploré jusqu'à une profondeur l, les chemins intéressants aux profondeurs inférieures ont déjà été explorés par le "branch and bound". Nous ne sommes donc plus obligés de réexplorer l'arbre pour les profondeurs inférieures. Par récursivité, on peut rapidement retrouver ces chemins.

Notre similarité devient :

$$S_{struct}(G, G^{'}) = \frac{1}{n_a} \sum_{i=1}^{|V|} \sum_{|h|=3}^{|h|=l} \max_{h_{v_i}, h^{'}} S_C(h_{v_i}, h^{'}) \tag{4.9}$$

avec

– n_a nombre d'appariements de chemins retournés

– l longueur maximale des chemins.

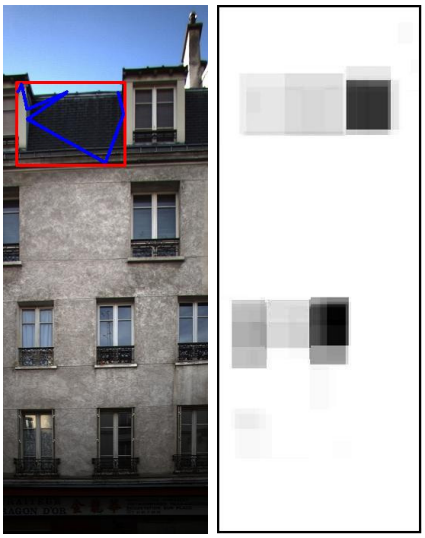
Nous cherchons ainsi pour un sommet v_i, le meilleur chemin commençant par celui-ci pour une taille 3 puis 4 ... jusqu'au chemin de taille l. Ainsi pour les meilleurs appariements, nous avons de la redondance. En effet, par exemple, si un chemin de taille |h| est un bon appariement alors la plupart du temps, nous constatons que le chemin de taille |h|+1 est une extension de ce chemin de taille |h|. Au contraire, si le chemin de taille |h| trouvé est un faux positif, alors le chemin de taille |h|+1 est souvent différent du chemin précédent.

Nous évaluons notre similarité (eq.(4.9)) sur une base composée de 200 images prises dans Paris dont 60 façades. Les images furent acquises à l'aide du système mobile STEREOPOLIS de l'IGN. Nous choisissons de faire varier la longueur de h de 3 à 8 ($3 \leq |h| \leq 8$). Pour un graphe requête fenêtre (imagette en haut à gauche), le système classe les images par ordre de similarité. Les meilleures images retournées (Fig.(4.11)) avec la similarité (eq.(4.9)) sont des façades contenant ce style de fenêtre.

FIGURE 4.11 – **Une fenêtre requête et les images retournées par ordre de similarité sur l'appariement de chemins de contours.** Les images sont classées de gauche à droite, puis de haut en bas. La façade au rang 1 contient la requête. Les façades signalées en rouge (rangs 5,6,13) sont des faux positifs. Les autres façades contiennent des fenêtres du même style que la requête.

4.2.1.2 Approche mélangant information contours et régions

Malgré l'adjonction du coefficient d'échelle $S_{e_i e'_i}$ (eq.(4.8)), le groupement des contours ne s'effectue pas toujours bien. Sur la figure (Fig.(4.12)), nous constatons que certains appariements entre chemins sont mauvais (Fig.(4.12 (a))) et contribue à une mauvaise détection de fenêtres (Fig.(4.12 (b))).



(a) (b)

FIGURE 4.12 – **Problème de la détection des fenêtres uniquement basée sur la similarité chemins. (a)** Une mauvaise région de chemin d'intérêt. **(b)** Accumulation des votes de régions d'intérêt uniquement basé sur la similarité chemin.

Nous devons diminuer l'importance de ces mauvais appariements lors du vote dans l'accumulation. Maintenant que nous obtenons des régions d'intérêt autour des meilleurs chemins, nous pouvons évaluer ces régions en utilisant des descripteurs régions. Nous pouvons donc introduir de l'information région (texture/couleur...) combinée à l'information portée par les contours. Nous proposons deux approches de combinaisons entre l'information régions et contours. La première approche consiste à sélectionner les meilleurs appariements de chemins uniquement sur l'information contours, puis d'ajouter l'information région sur ces meilleurs appariements. La deuxième approche consiste à sélectionner les meilleurs appariements de chemins à l'aide de l'information contours et régions. Une nouvelle similarité de chemins combinant les deux informations est proposée.

Approche contours, puis régions

Lors du calcul de similarité entre graphes (eq.(4.9)), nous cherchons, pour chaque chemin h_{v_i} du graphe requête, le chemin $h^{'}$ le plus similaire. Le chemin $h^{'}$ définit une RCI, que l'on peut comparer à la RCI définie par h_{v_i} . Nous proposons la similarité suivante qui combine les similarités des chemins (basées contours) et des régions :

$$S_{struct}(G, G^{'}) = \frac{1}{n_a} \sum_{i=1}^{|V|} \sum_{|h|=3}^{|h|=l} S_R(B_{h_{v_i}}, B_{h^{'}}) \max_{h_{v_i}, h^{'}} S_C(h_{v_i}, h^{'}) \tag{4.10}$$

avec les mêmes notations que l'équation (eq.(4.9)).

et l'accumulation de votes devient :

$$J_{h_{v_i}, h'}(x, y) = S_R(B_{h_{v_i}}, B_{h'}) S_C(h_{v_i}, h_{v_i}) \tag{4.11}$$

avec :
- $B_{h_{v_i}}$ (respectivement $B_{h'}$) la région d'intérêt englobant le chemin h_{v_i} (resp. $h^{'}$).
- $S_R(B_{h_{v_i}}, B_{h'})$ la similarité entre les deux régions d'intérêt $B_{h_{v_i}}$ et $B_{h'}$.
- $S_C(h_{v_i}, h^{'})$ similarité sur graphes de contours (eq.(4.8)).

Lors de nos tests de la similarité (Eq.(4.10)) (Fig.(4.13)), nous avons choisi une similarité texture entre les régions. Nous utilisons les histogrammes basés sur les ondelettes quaternioniques [CCB] et calculons la similarité à l'aide d'une fonction gaussienne et une distance du χ^2.

$$S_R(B_{h_{v_i}}, B_{h'}) = exp^{-\frac{1}{2}\chi^2(H, H^{'})^2}$$
$$\chi^2(H, H^{'}) = \sum_i \frac{(H[i] - H^{'}[i])^2}{H[i] + H^{'}[i]}$$

avec :

H et $H^{'}$ les histogrammes sur les régions $B_{h_{v_i}}$ et $B_{h'}$.

De plus, nous constatons sur la figure (Fig.(4.14)) que grâce à l'apport de la texture, lors de l'accumulation (eq.(4.11)) les mauvais appariements sont moins importants qu'avec l'accumulation uniquement sur la similarité chemins (eq.(4.7)).

FIGURE 4.13 – **Une requête et les images retournées par ordre de similarité basée sur les contours, puis sur les régions.** La façade au rang 1 contient la requête. Les façades signalées en rouge (rangs 15,17) sont des faux positifs. Les autres façades contiennent des fenêtres du même style que la requête.

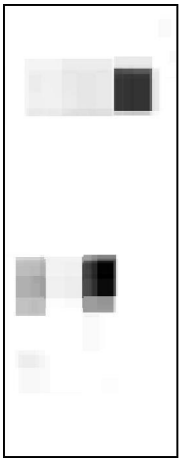


FIGURE 4.14 – **Accumulation des votes de régions d'intérêt sélectionnées sur la similarité chemins, puis combinées avec la similarité régions.**

Nous choisissons les meilleurs appariements uniquement avec la similarité sur chemins, puis nous atténuons ces appariements à l'aide d'une similarité régions. Cependant, certains appariements sélectionnés uniquement avec les contours

sont mauvais (contours de bruits) et il serait préférable de supprimer ces appariements au lieu de les atténuer. Pour celà, nous proposons de sélectionner les régions d'intérêts en combinant la similarité chemin et région lors de la sélection.

Approche contours et régions

Dans cette deuxième approche (Fig.(4.15)), nous proposons de rechercher les meilleurs appariements de chemins basés sur une similarité combinant contours et régions. Le choix du chemin ne se fait plus uniquement sur la suite de contours mais aussi sur la région définie par ces contours. Nous proposons de conserver la fonction de similarité définie par l'équation (eq.(4.9)) mais notre K_C devient :

$$\begin{aligned} S_{C_{CheReg}}(h_{v_i}, h') &= K_v(v_i, v'_0) \\ &+ \sum_{j=1}^{|h|} S_R(B_{h_{v_i}} B_{h'_{v'_j}}) K_e(e_j, e'_j) K_v(v_j, v'_j) \end{aligned} \tag{4.12}$$

FIGURE 4.15 – **Une requête et les images retournées par ordre de similarité conjointe sur les contours et les régions.** La façade au rang 1 contient la requête. Les autres façades contiennent des fenêtres du même style que la requête. Il n'y a plus de faux positifs dans le top 20.

De même pour l'accumulation, nous conservons la fonction de similarité entre RCI de l'équation (eq.(4.7)). Seule la valeur d'évaluation change $J_{h_{v_i},h'}(x,y)$. Dans notre cas, nous avons choisi un vote pondéré $J_{h_{v_i},h'}(x,y) = K_{C_{CheReg}(h_{v_i},h')}$.

Nous remarquons sur la figure (Fig.(4.16)) que grâce à l'apport de la texture, lors de la sélection des chemins, nous avons moins de mauvais appariements entre les fenêtres.

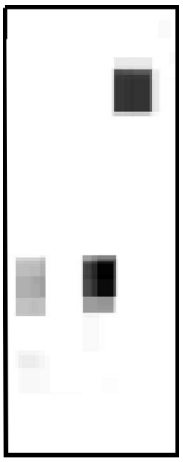


FIGURE 4.16 – **Accumulation des votes de régions d'intérêt sélectionnées sur la similarité combinant chemins et régions.**

4.2.1.3 Détection des fenêtres par apprentissage

En ce qui concerne la localisation des fenêtres, seules 2 ou 3 sont détectées sur la figure (Fig.(4.16)). Cela est dû au fait que chaque sommet du graphe requête ne vote que pour un seul chemin du graphe cible. Les chemins appariés dans le graphe cible peuvent se "répartir" sur plusieurs fenêtres (Fig.(4.9)), réduisant la détection de celles-ci mais aussi oblitérant la détection des autres. Afin d'augmenter le potentiel de généralisation de notre fonction de similarité et par là même trouver plus de fenêtres, nous proposons d'apprendre un ensemble $\mathbb{A}$ de fenêtres du même style. Pour cela, pour chaque image de la base, nous calculons la similarité avec chacune des images requêtes et ne conservons que les k plus grandes valeurs de similarité. La valeur de similarité par rapport à l'ensemble d'apprentissage $\mathbb{A}$ est :

$$S(G') = \frac{1}{k}\sum_{i=1}^{k} S_{struct}(G_i, G')$$

où

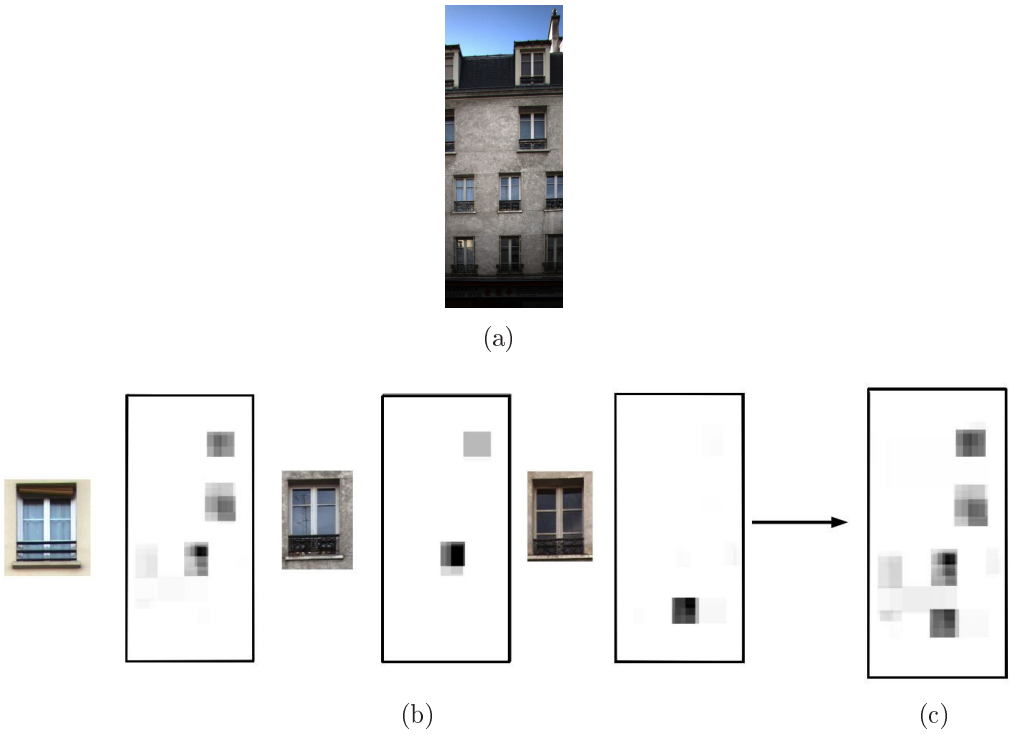
- $S_{struct}(G_i, G')$ est la similarité définie par équation (eq.(4.9)).
- $G_i \in \mathbb{A}_k$ graphe annoté.
- $\mathbb{A}_k$ ensemble des k plus proches voisins entre le graphe G et les graphes annotés avec la similarité (eq.(4.9)).

et l'accumulation devient :

$$J_{G'}(x, y) = \frac{1}{k} \sum_{i=1}^{k} y_i J_{G_i, G'}(x, y)$$

où $J_{G_i,G'}$ accumulation sur le graphe G' par rapport à la requête G (eq.(4.7)).

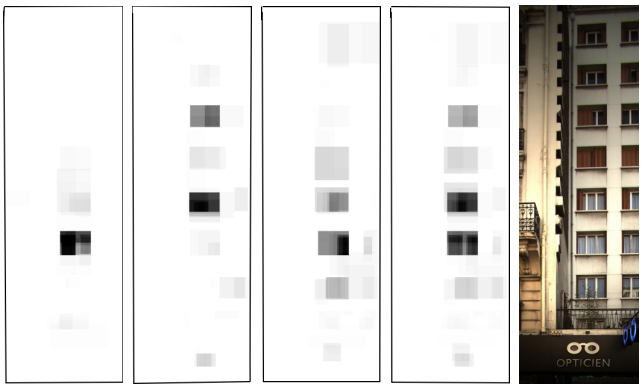
Les figures (Fig.(4.17) et Fig.(4.18)) montrent que la combinaison des requêtes permet de localiser plus de fenêtres.



(a)

(b) (c)

FIGURE 4.17 – **Accumulation sur une ensemble d'apprentissage de 3 fenêtres à 2 montants et un balcon. (b)** Accumulation obtenue avec 3 requêtes de même style. **(c)** Combinaison des 3 requêtes.



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

FIGURE 4.18 – **Accumulation sur une ensemble d'apprentissage de 3 fenêtres à 3 montants. (a) (b) (c)** Accumulation obtenue avec 3 requêtes de même style. **(d)** Combinaison des 3 requêtes.

Enfin, dans la section précédente, nous calculions une valeur de similarité entre un graphe requête de fenêtre et l'ensemble de la base comprenant des façades et des images négatives. Il est intéressant de noter que si nous souhaitons enrichir la recherche avec plusieurs graphes requêtes, les équations de similarité (eq.(4.10)) et (eq.(4.9)) n'étant plus symétriques car nous cherchons uniquement les chemins du graphe fenêtre dans la façade, nous ne pouvons plus utiliser le classifieur SVM. Nous choisissons donc de prendre un classifieur plus proche voisin pour classer les façades selon les différents graphes requêtes.

4.3 Dictionnaire de chemins et noyau incrémental

Nous avons défini précédemment un modèle afin de pouvoir comparer des chemins de contours entre eux et avons utilisé ce modèle pour comparer des graphes entre eux. Le problème de la comparaison entre graphes est sa forte complexité en temps de calcul. Afin de réduire la complexité de calcul, nous proposons de comparer les graphes à un dictionnaire de chemins. Dans cette partie, nous proposons d'utiliser la similarité de chemins dans le cadre des dictionnaires avec des noyaux adaptés.

Dans [GPS11], Gosselin *et al.* présentent un cadre théorique pour apprendre une fonction noyau lors d'une recherche interactive. Ce cadre n'a pas besoin d'hypothèses sur les descripteurs, l'exigence principale est une fonction d'évaluation qui n'a pas à satisfaire de propriétés mathématiques particulières. Ce cadre permet la combinaison des descripteurs de différents types, grâce à des fonctions d'évaluation adaptées à chaque type de descripteur. Une application de ce cadre est la capacité à construire un dictionnaire optimal lors de la recherche.

Nous proposons d'utiliser la stratégie de Gosselin *et al.* [GPS11], et de recueillir de façon dynamique un ensemble de chemins dans un dictionnaire, afin de projeter chaque graphe sur chaque mot (chemin) du dictionnaire. La mesure de similarité est calculée à l'aide d'un noyau incrémental. Nous avons fait le choix de travailler sur ce dictionnaire et ce noyau incrémental pour deux principales raisons. Premièrement, nous utilisons dans ce chapitre des noyaux qui vérifient les propriétés de Mercer, contrairement aux noyaux précédents qui ne vérifiaient pas toujours ces propriétés à cause de la recherche du maximum. Deuxièmement, nous espèrons réduire le temps de calcul. En effet, le nombre d'appariements entre un dictionnaire et un graphe est moindre qu'entre deux graphes. Et surtout nous utilisons un noyau incrémental pour la classification et ne calculons que ce qui est nécessaire pour mettre à jour le noyau précédent vers le nouveau.

Nous souhaitons développer un algorithme capable d'apprendre un dictionnaire adapté à la recherche interactive, c'est-à-dire capable à la fois de généraliser la classe que l'utilisateur souhaite trouver, mais aussi d'être le plus discriminant pour toutes les autres classes. Dans la figure (Fig.(4.19)), nous présentons le schéma de l'apprentissage que nous proposons d'utiliser dans ce chapitre. La principale différence avec les algorithmes usuels est que nous passons tous les processus liés au dictionnaire de l'étape hors ligne à l'étape en ligne. Un tel changement permet d'effectuer le calcul du dictionnaire avec l'information apportée par les étiquettes de l'utilisateur. Dans notre cas, nous utilisons les noyaux pour la classification et nous ne calculons ce qui est nécessaire pour mettre à jour le noyau précédent vers le nouveau.

Comme nous l'avons déjà vu, l'apprentissage actif est une approche qui s'intéresse au problème de la sélection des images à faire annoter par l'utilisateur. Les techniques qui lui sont apparentées, dites actives, vont déterminer les images qui, une fois annotées, donneront le meilleur résultat. Par exemple, un classifieur actif va sélectionner les images qui permettent de minimiser ensuite l'erreur de classification. Nous utilisons la stratégie détaillée par Gosselin et Cord [GC06].

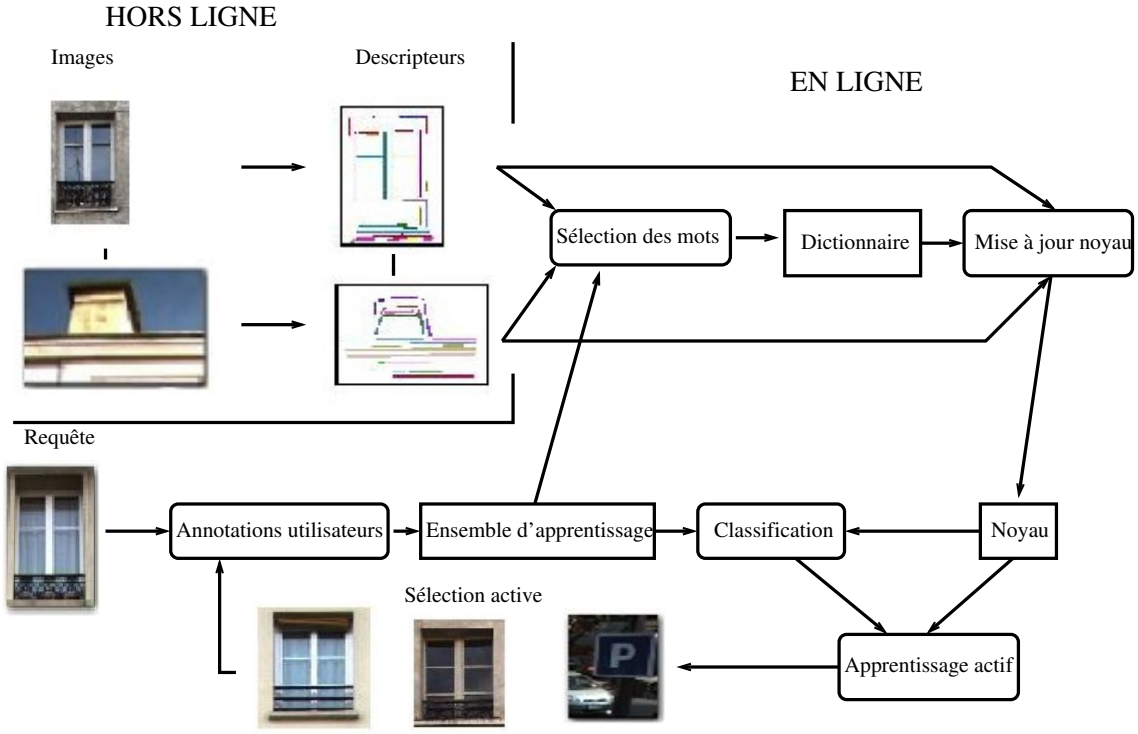


FIGURE 4.19 – **Algorithme proposé avec un dictionnaire dynamique**.

4.3.1 Théorie

Pour résoudre le problème de la complexité de calcul, le meilleur appariement entre chemins est trouvé à l'aide de l'algorithme de "séparation et évaluation". Toutefois, le problème de la comparaison de deux graphes G et G', par comparaison de chemins de même longueurs est encore un problème de grande complexité de calcul. Nous proposons de construire un dictionnaire de chemins afin de réduire le temps de calcul. L'idée est de comparer les graphes selon un dictionnaire $D_L = \{\hat{\mathbf{h}}_l\}_{l\in[1,L]}$ de chemins $\hat{\mathbf{h}}_l$. Nous désignerons les chemins dans le dictionnaire comme des mots. Le premier objectif est de construire le dictionnaire

$D_L = \{\hat{\mathbf{h}}_l\}_{l\in[1,L]}$ de chemins $\hat{\mathbf{h}}_l$ sélectionnés à partir des graphes des images de la base de données. L'avantage de la construction du dictionnaire avec des chemins à partir de graphes de la base est d'obtenir des résultats intéressants dans des sessions interactives qui vont effectuer la classification avec des ensembles d'apprentissage très faibles.

Au début d'une session de recherche, le dictionnaire D_0 est vide. Puis, à chaque retour d'évaluation, nous ajoutons des chemins dans le dictionnaire des graphes étiquetés. Pour chaque nouveau mot $\hat{\mathbf{h}}_\mathbf{l}$, nous construisons une fonction noyau mineur $k_{\hat{\mathbf{h}}_l}(G_i, G_j)$, qui calcule la similarité entre les graphes i et j par rapport au mot $\hat{\mathbf{h}}_l$. Nous sommons ensuite tous ces noyaux mineurs pour obtenir le noyau en fonction du dictionnaire courant D_L :

$$K_L(G_i, G_j) = S\left(\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i, G_j)\right) \tag{4.13}$$

avec L le nombre de mots dans le dictionnaire D_L et S une fonction quelconque telle qu'une fonction noyau.

Le calcul incrémental est ensuite simple :

$$K_{L+1}(G_i, G_j) = S\left(\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i, G_j) + k_{\hat{\mathbf{h}}_L+1}(G_i, G_j)\right) \tag{4.14}$$

Si $\mathcal{H}$ est l'espace des chemins, les noyaux mineurs $k_{\hat{\mathbf{h}}_l}(G_i, G_j)$ sont calculés à l'aide de la fonction d'évaluation $e_{\hat{\mathbf{h}}_l} : \mathcal{H} \to \mathbb{R}$ défini pour un chemin $\hat{\mathbf{h}}_l$ et d'une fonction δ basée sur une distance entre 2 réels :

$$k_{\hat{\mathbf{h}}_l}(G_i, G_j) = -\delta(e_{\hat{\mathbf{h}}_l}(G_i), e_{\hat{\mathbf{h}}_l}(G_j)) \tag{4.15}$$

La plus simple fonction d'évaluation $e_{\hat{\mathbf{h}}_l}$ vaut 1 si $\hat{\mathbf{h}}_l$ est dans G_i et 0 sinon. Cette formule peut être utilisée pour les dictionnaires de mots visuels, où nous vérifions seulement si les mots clés $\hat{\mathbf{h}}_l$ appartiennent à l'image G_i (par exemple pour les graphes étiquetés).

La fonction δ compare l'évaluation des graphes des images i et j en fonction de la caractéristique $\hat{\mathbf{h}}_l$. Cette fonction doit être choisie de telle sorte que la fonction K_L soit une fonction noyau. Ainsi la fonction $e_{\hat{\mathbf{h}}_l}$ n'a pas besoin de satisfaire de propriétés mathématiques nécessaires aux fonctions noyaux.

4.3.2 Deux nouveaux noyaux sur graphes

Dans l'équation (eq.(4.15)), nous avons besoin d'une fonction $e_{\hat{\mathbf{h}}_l}(G_i)$ qui évalue à quel point un chemin du dictionnaire $\hat{\mathbf{h}}_\mathbf{l}$ est similaire à un chemin de G_i. Cette fonction peut simplement être la similarité maximale entre $\hat{\mathbf{h}}_\mathbf{l}$ et un chemin de G_i :

$$e_{\hat{\mathbf{h}}_l}(G_i) = \max_{\mathbf{h} \in G_i} K_C(\hat{\mathbf{h}}_l, \mathbf{h}) \tag{4.16}$$

K_C (eq.(4.6)) est la similarité entre deux chemins.

Nous définissons la fonction de similarité S (eq.(4.13)) afin de construire deux nouveaux noyaux. Contrairement au noyau sur graphes K_{struct} (eq.(4.1)), qui ne vérifie pas toutes les propriétés de Mercer à cause de la recherche du maximum, les noyaux définis dans cette section sont des noyaux qui vérifient ces propriétés :

– Noyau triangulaire

$$\begin{aligned} K_L^{Tri}(G_i, G_j) &= \sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i, G_j) \\ &= -\sum_{l} \delta(e_{\hat{\mathbf{h}}_l}(G_i), e_{\hat{\mathbf{h}}_l}(G_j)) \\ &= -d(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{4.17}$$

avec δ une fonction de deux réels et d une distance entre deux vecteurs.

– Noyau gaussien

La méthode peut aussi être utilisée pour construire un noyau gaussien de manière incrémentale.

$$\begin{aligned} K_L^{Gau}(G_i, G_j) &= \exp\left(\frac{1}{2\sigma^2} \sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i, G_j)\right) \\ &= \exp\left(-\frac{1}{2\sigma^2} \sum_{l=1}^{L} \delta(e_{\hat{\mathbf{h}}_l}(G_i), e_{\hat{\mathbf{h}}_l}(G_j))\right) \\ &= \exp\left(-\frac{1}{2\sigma^2} d(\mathbf{x}_i, \mathbf{x}_j)\right) \end{aligned} \tag{4.18}$$

Les noyaux (eq.(4.17) et eq.(4.18)) peuvent être utilisés avec plusieurs distances. Par exemple, les distances suivantes :

$$d_{L^1}(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1$$

$$d_{\chi^1}(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{n} \left|\frac{x_i - y_i}{x_i + y_i}\right| \tag{4.19}$$

$$d_{L^2}(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2$$

$$d_{\chi^2}(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$$

4.4 Conclusions

Nous avons montré dans ce chapitre l'intérêt d'exploiter les contours pour effectuer la recherche d'objets dans une base d'images mais aussi pour localiser ces objets dans l'image. La principale difficulté est d'obtenir le regroupement des contours appartenant à l'objet recherché. Pour cela, nous avons proposé un cadre sur la similarité de chemins de contours basé sur des noyaux robustes aux changements d'échelle.

Cependant, il a été nécessaire d'adjoindre une information issue de la radiométrie des images pour renforcer la similarité des ensembles de contours. Des similarités combinant informations contours et régions délimitées par ces contours ont été proposées. Intégrées dans des classifieurs de type SVM ou kppv, les similarités proposées permettent de retrouver (et de classer) des images comme contenant un objet particulier décrit par l'exemple. Le noyau sur chemins proposé permet en outre une localisation de l'objet ou différentes occurences de l'objet dans l'image.

De plus, pour réduire le problème de la complexité de calcul, nous avons proposé un nouveau noyau sur des graphes basés sur un dictionnaire de chemins. La mesure de similarité est calculée à l'aide d'un noyau incrémental basé sur une fonction d'évaluation. Dans notre cas, cette fonction d'évaluation est la similarité sur chemins de contours proposée dans notre modèle de mise en correspondance de chemins de contours. L'avantage de ce noyau est sa propriété incrémentale qui permet de calculer uniquement ce qui est nécessaire pour mettre à jour le noyau précédent vers le nouveau.

L'algorithme de "séparation et évalutation" (voir chapitre 5) employé avec ces nouveaux noyaux permet de trouver des solutions exactes ou approchées au problème de l'appariement de graphes avec des graphes ou des sous-graphes d'autres images. Le problème majeur de ces noyaux est la forte complexité de calcul sur des graphes de 40 sommets en moyenne pour les graphes de fenêtres et des graphes de 500 sommets pour les façades. Nous verrons dans le chapitre suivant l'intéret de l'algorithme de "séparation et évalutation" pour trouver une approximation du maximum et donc réduire le temps de calcul en limitant les appariements possibles. Enfin, une évaluation de nos noyaux en fonction des différents paramètres (longueur des chemins, poids échelle, poids orientation...) sera menée dans le chapitre 5.

Chapitre 5

Implémentation et résultats

Dans ce chapitre, nous discutons de l'implémentation et de la représentation par arbre de recherche des noyaux sur graphes. Cette représentation nous permet d'utiliser l'algorithme de "séparation et évaluation" afin de réduire la complexité des calculs. Puis nous évaluons le modèle mis en place dans le chapitre 4 pour comparer des graphes de contours.

5.1 Représentation par arbre de recherche

Il existe différentes manières d'implémenter les techniques par chemins aléatoires, et parmi celles-ci nous avons choisi l'algorithme de "séparation et évalutation". Cet algorithme permet de calculer rapidement une valeur approximée de la similarité ce qui permet d'assurer par exemple que deux graphes globalement similaires auront une valeur de similarité élevée. De plus il n'est pas nécessaire d'énumérer l'ensemble des chemins pour pouvoir obtenir un résultat, ce qui permet de réduire les coûts mémoire. Enfin, cet algorithme se parallélise très bien. Nous présentons la représentation par arbre de recherche qui permet par la suite d'optimiser la recherche des meilleurs appariements. Puis nous introduisons l'algorithme de "séparation et évaluation" sur nos noyaux sur graphes.

La complexité de calcul dépend du nombre de chemins de $H(G)$, de la fonction de similarité et du noyau sur chemins. $K_{Kashima}$ est un noyau qui nécessite une énumération exhaustive de toutes les comparaisons possibles entre chemins (eq.(3.5)), car il effectue la somme de toutes les similarités entre chemins. Si une solution incomplète est suffisante, le seul moyen de réduire le calcul avec ce noyau est lié à la longueur des chemins. Au contraire, pour calculer le noyau K_{max} (eq.(3.6)), une sous-partie de chemins peut être suffisante. La recherche du maximum peut être facilement obtenue par l'algorithme de "séparation et évaluation"

(plus connu sous "branch and bound" [Cla97]). Cette solution incomplète par élagage de l'arbre de recherche est souvent suffisante. K_{Suard} (eq.(3.7)), K_{Lebrun} (eq.3.8) et nos noyaux proposés peuvent ainsi être calculés sans effectuer toutes les comparaisons possibles de chemins.

5.1.1 L'arbre de recherche

Une propriété intéressante des chemins d'un graphe est l'aspect récursif de son parcours. Un arbre de recherche est utilisé pour représenter toutes les comparaisons possibles entre chemins. Cette représentation et les propriétés de récursivité nous permettent d'améliorer la recherche des meilleurs appariements et de réduire les temps de calcul à l'aide de l'algorithme de "séparation et évaluation".

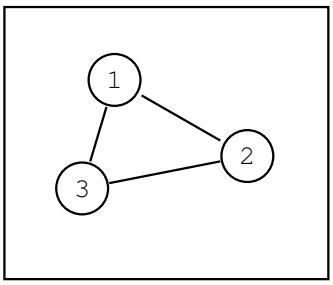
Pour deux graphes $G = (V, E)$ et $G' = (V', E')$, notre arbre de recherche (Fig.(5.1)) est composé :

- d'une racine.
- de noeuds : chaque noeud $n = (v, v')$ représente un appariement entre deux sommets des graphes $v \in G$ et $v \in G'$.
- d'une branche entre deux noeuds $n_1 = (v_1, v'_1)$ et $n_2 = (v_2, v'_2)$ signifie qu'il existe un arc $e_{1,2}$ entre v_1 et v_2, et un arc $e'_{1,2}$ entre v'_1 et v'_2.

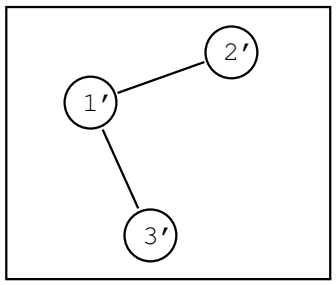
Un exemple d'arbre de recherche est montré sur la figure (Fig.(5.1)). Un chemin dans l'arbre de recherche commence par la racine ne contenant aucune information puis un chemin est une succession d'appariements de sommets des graphes G et G'. Un parcours entre la racine et une feuille correspond à un appariement entre un chemin de G et un de G'. Le principal intérêt de la recherche du maximum dans l'arbre de recherche est que ce dernier n'a pas besoin d'être complètement construit, nous pouvons "élaguer" durant sa construction.

5.1.2 Application de l'algorithme d'"évaluation et séparation" sur les noyaux sur chemins

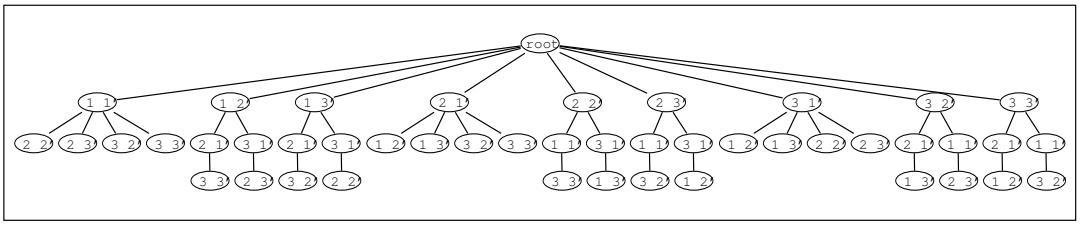
L'algorithme de "séparation et évalutation" [Cla97] a pour objectif de trouver de manière optimale la valeur maximum d'évaluation. Il est spécialement adapté pour résoudre la recherche du maximum de fonctions dont les limites peuvent être prévues sur un sous-ensemble déterminé (dans notre cas, un sous-ensemble de chemins).



(a) Un exemple de graphe G



(b) Un exemple de graphe G'



(c) L'arbre de recherche correspondant, sans cycle et boucle.

FIGURE 5.1 – **Un exemple d'arbre de recherche.** Chaque parcours depuis la racine vers une feuille est un appariement possible entre deux chemins de G et G'. Par exemple, le chemin correspondant à la comparaison des chemins (213) et $(3'1'2')$ est $(root) \rightarrow (23') \rightarrow (11') \rightarrow (32')$.

Définition 5.1.

Algorithme de séparation et d'évaluation (branch and bound) (Clausen [Cla97]) : *est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète. C'est une méthode d'énumération implicite : toutes les solutions possibles du problème peuvent être énumérées mais, l'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions. Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées.*

5.1.2.1 Récursivité des noyaux

Dans notre cas, la fonction à limiter ("bound") est la fonction noyau K_C. Cette fonction peut être calculée récursivement. Ajouter un noeud à un chemin de l'arbre revient à rajouter un appariement de 2 sommets et de 2 arêtes du graphe G et G'. En pratique, soient $h_i \in H(G)$ et $h'_i \in H(G')$, en ajoutant un noeud v et une arête e à h_i nous obtenons h_{i+1} (respectivement v' et e', nous avons h'_{i+1}). Le noeud est ajouté à l'arbre comme un prolongement au chemin représentant

l'appariement entre (h_i, h'_i).

Pour être efficace, le calcul de la similarité $K_C(h_{i+1}, h'_{i+1})$ doit être calculé à partir du calcul de $K_C(h_i, h'_i)$. Tous les noyaux sur chemins présentés dans le chapitre précédent possèdent cette propriété.

La formule récursive s'écrit :

$$K_C(h_{i+1}, h'_{i+1}) = K_C(h_i, h'_i) + K_e(e, e')K_v(v, v')$$

5.1.2.2 Notre implémentation

Dans les noyaux proposés, nous cherchons le meilleur appariement, cependant il existe différentes façons d'explorer l'arbre à la recherche de ce maximum. Nous devons trouver une manière astucieuse d'évaluer les différentes branches afin d'éviter d'explorer la totalité de l'arbre. L'évaluation d'un noeud de l'arbre de recherche a pour but de déterminer l'optimum dans l'ensemble des solutions réalisables associé au noeud en question ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, que la solution optimale n'est pas sur une branche issue de ce noeud). Lorsqu'un tel noeud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions.

Lors de la construction de l'arbre, nous développons les 2 premiers niveaux après la racine (Fig.(5.1)) :

- A la profondeur 1, il s'agit des appariements possibles entre les sommets du graphes G et G'.

- Puis à la profondeur 2, nous avons tous les appariements possibles entre les arêtes et noeuds de G et G'. A cette étape nous connaissons le meilleur appariement arêtes-sommets (selon notre similarité sommets et similarité arêtes) et notons $S_{max}(ev, e'v')$.

$$S_{max}(ev, e'v') = \max_{\substack{e \in E, e' \in V \\ v \in V, v' \in V'}} K_e(e, e') \times K_v(v, v')$$

- Ensuite, si nous souhaitons trouver le meilleur appariement de chemins de taille l, nous devons développer l'arbre jusqu'à la profondeur $l + 2$. Pour des raisons d'espace mémoire (Sakarovitch [Sak84]), nous sommes partis sur une stratégie "profondeur d'abord". A l'aide du parcours en profondeur, nous cherchons donc la première feuille à la profondeur $l+2$ (feuille

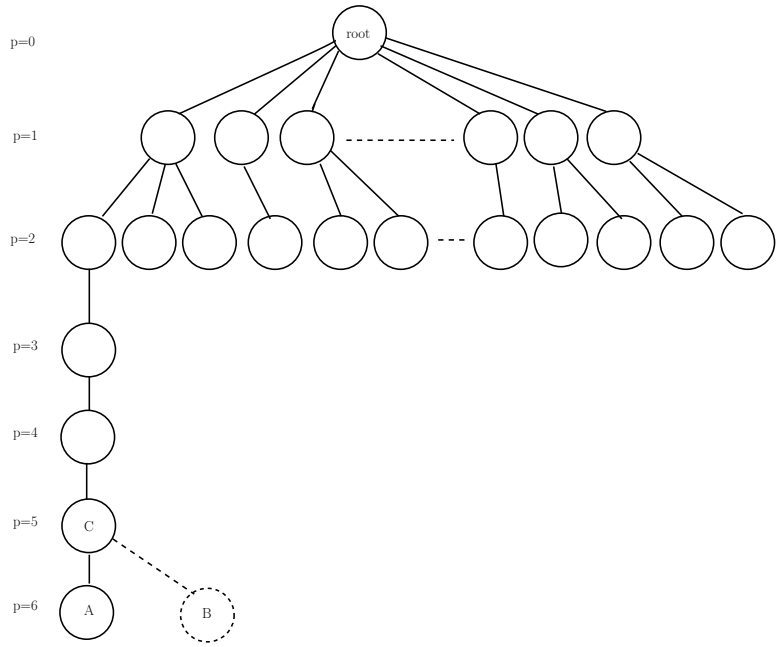


FIGURE 5.2 – **Construction de l'arbre de recherche.** Quelles sont les conditions permettant d'ajouter la feuille B au chemin se terminant en C ? Le chemin menant à B a t-il une similarité plus importante que celui menant à A. La feuille B est explorée si la valeur de similarité portée sur C plus la valeur théorique du meilleur appariement arêtes-noeuds est supérieure à la valeur de similarité maximum déjà trouvée, ici sur le chemin se terminant en A.

A sur l'exemple de la figure (Fig.(5.2)). Cette feuille est valuée par la similarité $K_C(h_l, h'_l)$. Cette première valeur est considérée pour l'instant comme la valeur maximum $V_{maxTrouv}$. Puis nous remontons à son noeud père qui posssède une valeur de similarité $K_C(h_{l-1}, h'_{l-1})$. Nous évaluons alors les branches possibles qui peuvent être développées à partir de ce noeud. L'évaluation est la suivante : connaissant la valeur maximum trouvée et la valeur du noeud courant V_{noeudC} et sa profondeur l_{noeudC} (avec $l_{noeudC} < (l+2)$), si l'ajout du meilleur couple arêtes-noeuds peut améliorer le maximum déjà trouvé, alors nous devons explorer cette branche. Nous devons calculer le maximum théorique que nous allons obtenir en explorant cette branche.

Ce maximum théorique est :

$$V_{maxTheo} = V_{noeudC} + ((l+2) - l_{noeudC}) \times S_{max}(ev, e'v')$$

Nous explorons la branche alors si :

$$V_{maxTheo} > V_{maxTrouv}$$

Nous donnons quelques exemples sur les figures (Fig.(5.2) et Fig.(5.3)).

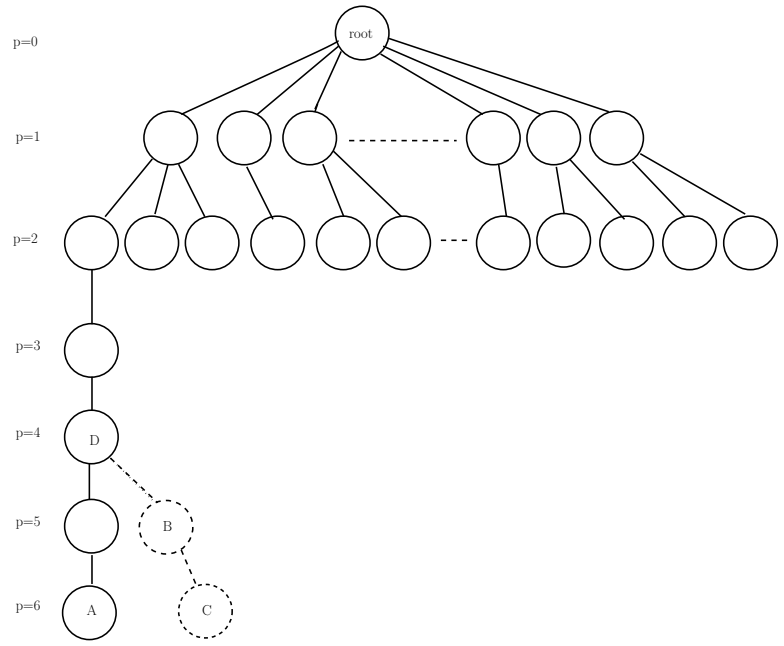


FIGURE 5.3 – **Construction de l'arbre de recherche.** Devons-nous explorer au niveau de D la branche en pointillé ? L'ajout des noeuds B et C vont-ils permettre d'avoir une meilleure similarité que le chemin menant A ? Le noeud B est exploré si la valeur de similarité portée sur C plus 2 fois la valeur théorique du meilleur appariement arêtes-noeuds est supérieure à la valeur de similarité maximum déjà trouvée, ici sur le chemin se terminant en A.

5.2 Evaluation

Dans le chapitre 2, nous avons comparé notre algorithme d'extraction d'hypothèses de fenêtres à celui de Lee et Nevatia. Si notre algorithme est le plus performant, il reste beaucoup de faux-positifs. Nous avons donc décidé de représenter les hypothèses de fenêtres par des graphes de régions et de trouver une fonction d'évaluation afin de discriminer les fenêtres des faux-positifs (*cf.* chapitre 4). Dans ce chapitre, nous allons évaluer la pertinence de la fonction d'évaluation proposée et ses différents paramètres.

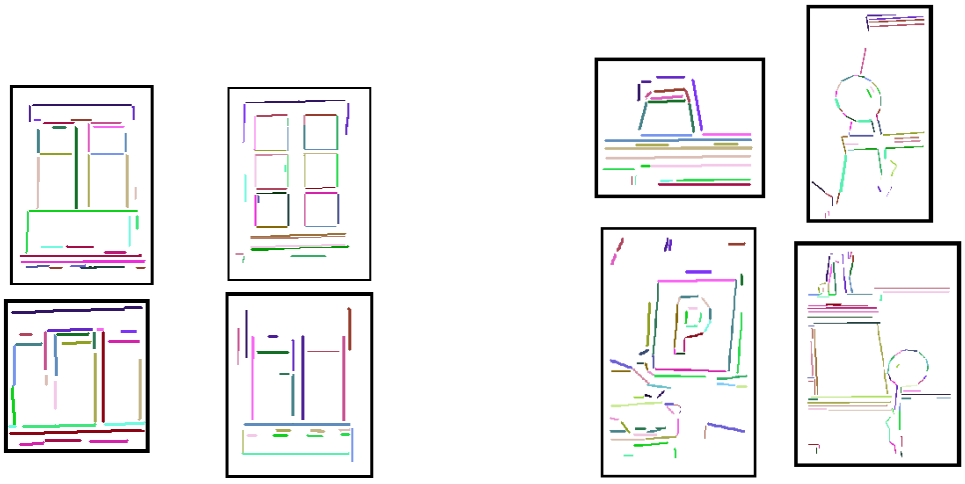
(a) Fenêtre (b) Négatifs

FIGURE 5.4 – Exemples d'hypothèses de notre base : 70 fenêtres et 150 négatifs

Nous avons extrait des hypothèses de fenêtres et formé une base de 220 images avec 70 fenêtres et 150 fausses alarmes (Fig.(5.4)). Chaque image est alors représentée par son graphe de segments de contours (Fig.(5.5)) comme détaillé dans le chapitre 2. La base est constituée de graphes de contours dont le nombre de noeuds est compris entre 10 et 133, et un nombre d'arêtes moyen de 80.

5.2.1 Protocole de simulation interactive

Nous simulons une session de recherche interactive pour un objet ou une catégorie, dans notre cas une fenêtre de la base de données. Nous évaluons plusieurs noyaux en simulant un grand nombre de sessions de recherche. Pour chaque



(a) Contours des fenêtres (b) Contours des négatifs

FIGURE 5.5 – Contours des images de la figure (Fig.(5.4))

session, une fenêtre est sélectionnée aléatoirement et annotée comme positive. Un premier classement de la base est alors effectué, seulement basé sur la similarité sur graphes. Ensuite quelques images sélectionnées par l'actif (Gosselin *et al.* [GCPF07b]) sont annotées selon la présence ou non d'une fenêtre (comme positif ou négatif). Le Séparateur à Vaste Marge est alors entraîné avec cet ensemble d'exemples annotés, conduisant ainsi un meilleur classement de la base de données. Le processus de classification est alors répété avec le même principe de sélection par *bouclage de pertinence* et de classification. La qualité moyenne du classement est mesurée à chaque retour de pertinence des annotations à l'aide du critère de précision. La simulation des sessions de recherche est répétée 50 fois et les résultats sont moyennés.

5.2.2 Evaluation des noyaux sur chemins pour la recherche d'images

5.2.2.1 Evaluation des pondérations dans notre noyau

Nous avons comparé nos noyaux avec et sans les différents poids proposés dans le chapitre 4.

Nous observons sur la figure (Fig.(5.6)) un écart important entre K_{struct} sans poids et K_{struct} avec facteur échelle $S_{e_i e'_i}$ et/ou avec le poids d'orientation O_{ij}. Ces poids apportent de l'information structurelle et améliorent ainsi le groupement perceptuel de l'ensemble des contours. Nous remarquons qu'avec seulement 10 labels et notre noyau K_{struct} avec les 2 poids, la précision moyenne est au-dessus de 90%.

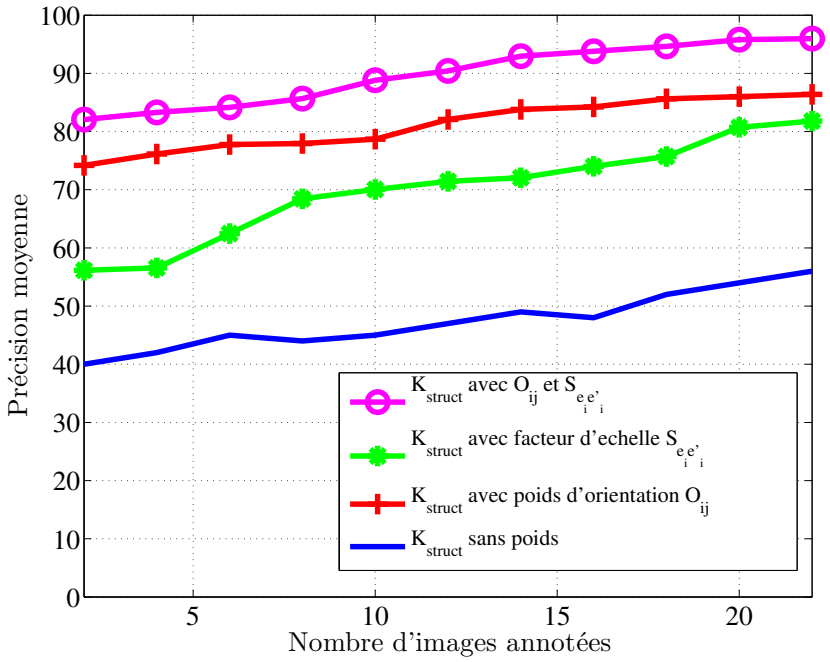


FIGURE 5.6 – Comparaison des différents noyaux sur chemin avec ou sans les différents poids : facteur d'échelle et poids d'orientation des contours. $|h| = 5$

5.2.2.2 Evaluation de la taille des chemins

K_{struct} avec son noyau mineur K_C (eq.(4.6)) compare les chemins avec une longueur fixée. Comme nous l'avons noté dans la section 5.1, plus le chemin

est long, plus nous devons explorer l'arbre de recherche. Nous devons choisir la meilleure longueur de chemins afin que les chemins décrivent pleinement la structure de l'objet le plus rapidement possible. Nous avons testé des longueurs de chemins entre 3 et 8 (Fig.(5.7)). Avec un chemin de longueur 3, nous ne satisfaisons pas pleinement la structure de l'objet et ne profitons pas de la structure du graphe. Nous notons que plus le chemin est long, plus nous gagnons en précision moyenne mais nous perdons en temps de calcul (Tab.(5.1)). Par la suite, nous avons décidé de travailler sur des chemins de longueur $|h| = 5$ qui donne un bon compromis entre temps de calcul et qualité des résultats.

Longueur	3	4	5	6	7	8
Temps moyen (millisecondes)	55	61	67	77	84	94

TABLE 5.1 – Temps de calcul moyen (en millisecondes) pour calculer la similarité entre un graphe G de 30 contours et un autre graphe, pour différentes longueurs de chemins.

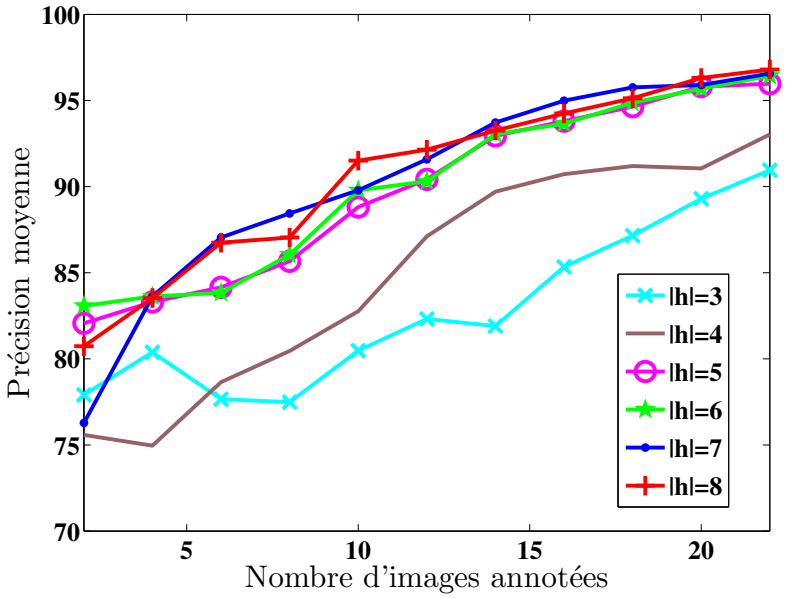


FIGURE 5.7 – Comparaison des résultats pour des chemins de taille entre 3 et 8 dans notre K_C

5.2.2.3 Evaluation de notre noyau par rapport à d'autres méthodes

Nous avons aussi comparé à un noyau sur sac K_{bag} de $\mathrm{B}_i = \{b_{ri}\}_r$ où r est une région MSER décrite par un vecteur SIFT b_{ri} [WCG03] :

$$K_{bag}(B_i, B_j) = \frac{1}{|B_i|}\sum_r \left(\sum_s k(b_{ri}, b_{sj})^q\right)^{\frac{1}{q}} + \frac{1}{|B_j|}\sum_s \left(\sum_r k(b_{ri}, b_{sj})^q\right)^{\frac{1}{q}}$$

$$\text{avec } k(b_r, b_s) = exp\left(\tfrac{1}{2\sigma^2} d_{\chi^2}(b_r, b_s)\right)$$

avec $q = 2$

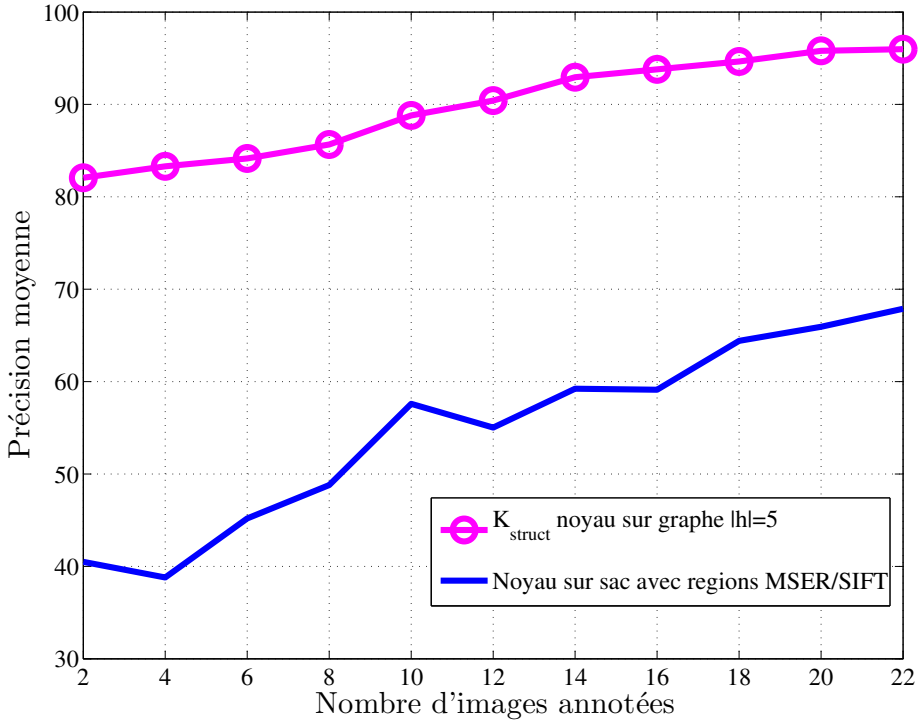


FIGURE 5.8 – Comparaison entre notre noyau sur graphe et le noyau sur sac de région MSER.

Notre noyau K_{struct} (eq.(4.1) et eq.(4.6)) donne de meilleurs performances que le noyau sur sacs K_{bag} (Fig.(5.8)). K_{bag} est moins précis que K_{struct} étant

donné que le noyau sur sacs ne tient pas en compte la relation spatiale entre les régions et ne décrit donc pas la structure des objets.

Par ailleurs, nous nous sommes comparés au noyau K_{Lebrun} (eq.(3.8)) proposé par Lebrun *et al.* [LPFG08]. Ce noyau permet de réduire le nombre de chemins à comparer. En effet, lors de la recherche du meilleur appariement, le nombre de chemins dans G' est réduit, en ne gardant dans l'ensemble de recherche que les chemins qui commencent par le noeud $v'_i \in V', v'_i = s(v_i)$ le plus similaire à v_i.

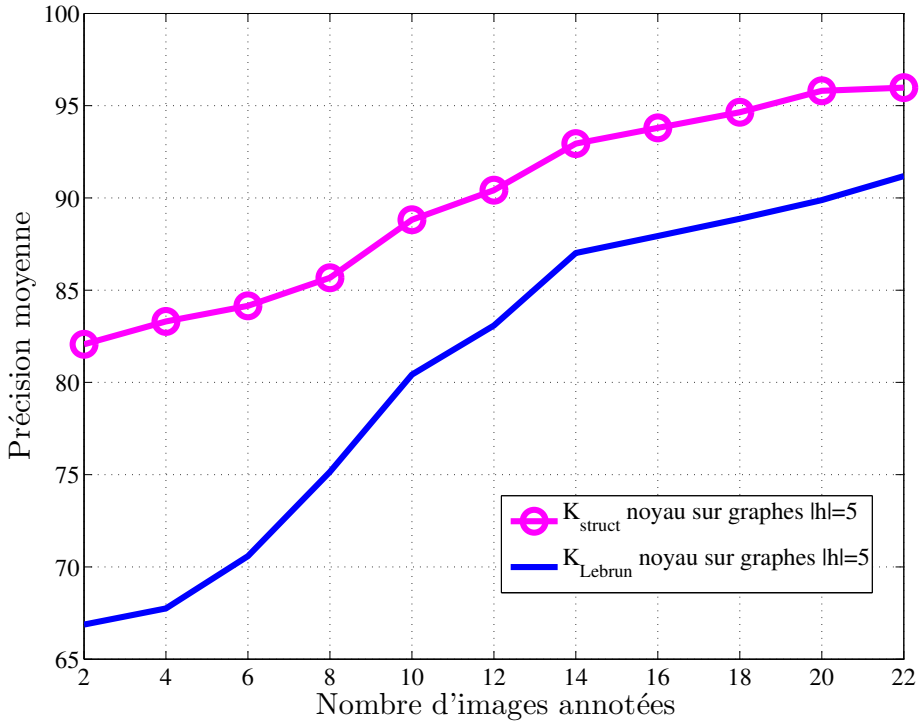


FIGURE 5.9 – Comparaison entre notre noyau sur graphes K_{struct} et le noyau sur graphes K_{Lebrun}

Nous constatons sur la figure (Fig.(5.9)) que notre noyau est plus performant que le noyau proposé par Lebrun. La contrainte sur les sommets de départ des chemins proposée par Lebrun ne permet pas dans notre cas de trouver une approximation du meilleur appariement entre chemins. Dans le cas de graphes sur régions, K_{Lebrun} est efficace car les régions possèdent de fortes caractéristiques et donc sont faciles à différencier. Dans notre cas de graphes sur contours, les images

possèdent beaucoup de contours verticaux et horizontaux, ainsi si nous devons choisir au départ le sommet $v'_i \in V', v'_i = s(v_i)$ le plus similaire à v_i, nous avons de multiples possibilités. Parmi cet ensemble de multiples possibilités, si nous choisissons le premier sommet v'_i trouvé, alors nous nous restreignons toujours au même sous-graphe. Les autres possibilités de l'ensemble des sommets de départ ne sont pas prises en compte et nous ne pourrons alors pas explorer les nombreux chemins du graphe. De plus, si nous choisissons un noeud v'_i pris au hasard dans l'ensemble des noeuds les plus similaires, nous n'avons pas nécessairement pris de bons chemins. Sur la figure (Fig.(5.9)), nous avons choisi de sélectionner les noeuds v'_i au hasard dans l'ensemble des noeuds les plus similaires. Le résultat confirme les hypothèses précédentes, le choix des chemins n'est pas forcément optimal.

5.2.3 Recherche rapide à l'aide de dictionnaire de chemins

Afin d'aborder le problème de la complexité de temps de calcul, la recherche du meilleur appariement est plus efficace avec l'algorithme de "séparation et évaluation". Cependant le problème de comparaison de 2 graphes G et G' par comparaison de chemins de même longueur renvoie au problème de complexité de temps de calcul. Nous avons proposé alors dans le chapitre 4 une nouvelle méthode basée sur un dictionnaire de chemins de contours afin de réduire le temps de calcul.

5.2.3.1 Dictionnaire de chemins

Avant toute session de recherche, nous avons extrait les chemins les plus représentatifs de chaque graphe G_i avec une longueur fixe. Pour être sûr de couvrir l'ensemble du graphe, nous avons considéré que chaque noeud du graphe était le début d'un chemin à extraire. L'ensemble des chemins issus de $G_i = (V_i, E_i)$ contient donc au plus $|V_i|$ chemins (les chemins très similaires peuvent être représentés par un seul chemin). Nous avons aussi montré dans le chapitre 4 que les chemins dont les sommets successifs (qui représentent des segments de contour) ne sont pas parallèles proposent une meilleure structure de l'objet. Pour ces raisons, le choix du meilleur chemin débutant par le sommet v_j est le chemin $h = (v_j, ...)$ qui maximise :

$$\max_{h_{v_j} \in H_{v_j}(G)} \sum_{k=j}^{j+|l} \sqrt{\frac{1}{2}(1 - \langle v_k, v_{k+1} \rangle)} \text{ avec } l \text{ la longueur du chemin.}$$

Soit $H_{v_j}(G)$, l'ensemble des chemins qui commencent par le sommet v_j. Nous cherchons donc dans cet ensemble le "meilleur" chemin simple dont les sommets successifs sont les plus différents possible les uns des autres. Les chemins avec cycles ou boucles ne sont pas acceptés. Seuls les chemins des graphes annotés positivement seront ajoutés dans le dictionnaire.

Dans cette partie, nous testons la méthode avec un noyau gaussien (eq.(4.18)) et différentes distances (eq.(4.19)) sur des appariements de chemins de taille 5 (Fig.(5.10)). La même comparaison est réalisée avec le noyau triangulaire (eq.(4.17) et Fig.(5.11)). Avec le noyau gaussien, les meilleures performances sont obtenues avec les distances χ^1 et L_1 , tandis que les différentes distances ne changent rien avec le noyau triangulaire.

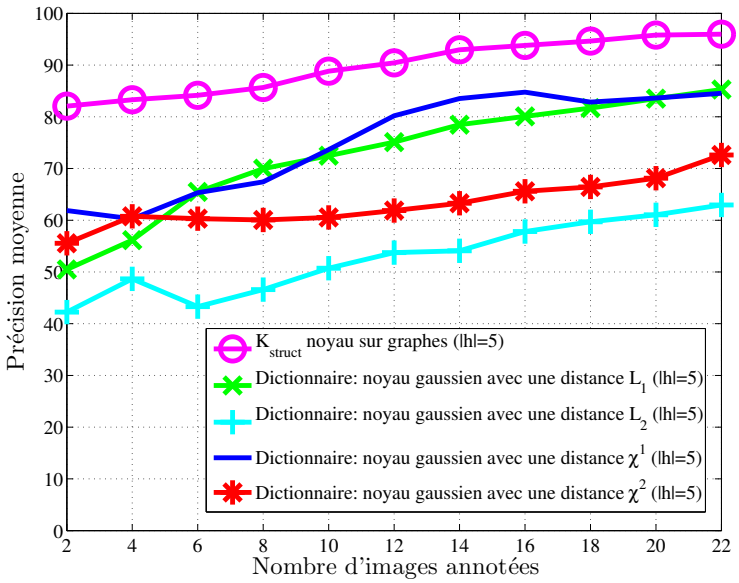


FIGURE 5.10 – Comparaison des noyaux gaussiens avec différentes distances sur le dictionnaire dynamique et K_{struct}.

Pour peu de labels, le noyau gaussien (excepté avec la distance L_2) est meilleur que le noyau triangulaire. Tous ces noyaux sont moins précis que le noyau K_{struct}, mais notons que la précision moyenne est déjà élevée pour 2 labels. Si nous comparons avec le noyau K_{struct} (eq.(4.1)), nous obtenons une précision moyenne de 80%. Celle-ci est de 50% avec le noyau triangulaire et de 62% avec le noyau gaussien. Cependant le principal intérêt du dictionnaire dynamique est son efficacité en temps de calcul (Tab.(5.2)). En moyenne, la similarité entre un graphe

de 26 noeuds avec un autre graphe est obtenu en 28 millisecondes avec K_{struct} et en 4 millisecondes pour le dictionnaire dynamique. Pour un graphe de 120 noeuds, le temps de comparaison passe de 362 millisecondes à 34 millisecondes. Le dictionnaire dynamique est donc environ 10 fois plus rapide.

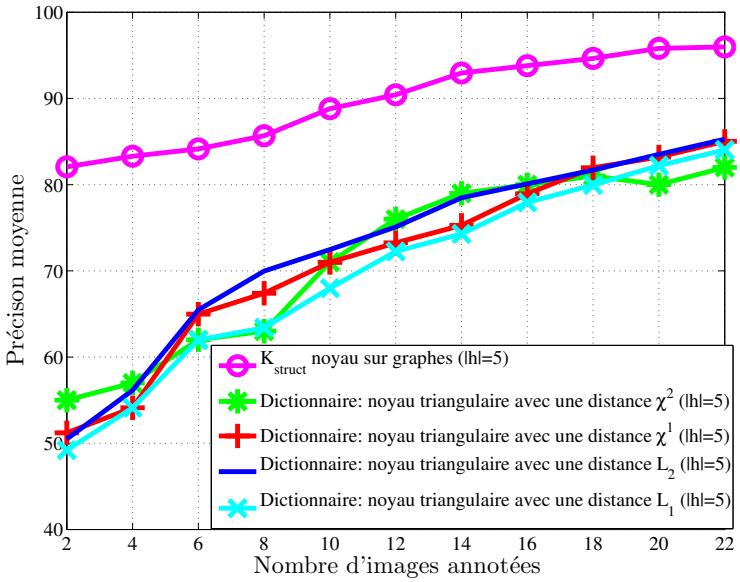


FIGURE 5.11 – Comparaison des noyaux triangulaires avec différentes distances sur le dictionnaire dynamique et K_{struct}.

	Nombre de noeuds dans le graphe G			
	26	30	40	124
K_{struct}	28	68	132	362
Dictionnaire avec noyau triangulaire	3	4	6	34
Dictionnaire avec noyau gaussien	4	5	8	37

TABLE 5.2 – Temps de calcul moyen (en millisecondes) pour le calcul de similarité de graphes entre un graphe G et un autre graphe pour des chemins de taille $|h| = 5$.

5.2.3.2 Sacs de chemins

Dans le cadre du calcul du dictionnaire, pour chaque graphe G_i nous sélectionnons des chemins qui maximisent :

$$\max_{h_{v_j} \in H_{v_j}(G)} \sum_{k=j}^{j+|l|} \sqrt{\frac{1}{2}(1 - \langle v_k, v_{k+1} \rangle)}.$$

avec $H_{v_j}(G)$, l'ensemble des chemins simples qui commencent par le sommet v_j. Nous cherchons donc dans cet ensemble le "meilleur" chemin simple dont les sommets successifs sont les plus différents possible les uns des autres.

Ces chemins peuvent être ensuite injectés dans le dictionnaire puis nous comparons chaque graphe à ce dictionnaire. Dans cette partie, nous souhaitons réduire encore le nombre de chemins à comparer et proposons de représenter chaque image comme un sac B_i de chemins $h^i_{v_j}$ non ordonnés.

Pour un graphe $G_i = (V_i, E_i)$ est associé un sac B_i tel que :

$$B_i = \left\{ h^i_{v_j} | \forall v_j \in V_i, \exists! h_{v_j} = \max_{h_{v_j} \in H_{v_j}(G_i)} \sum_{k=j}^{j+|l|} \sqrt{\frac{1}{2}(1 - \langle v_k, v_{k+1} \rangle)} \right\}$$

avec l la longueur du chemin.

Nous avons donc un sac de chemins avec $|V_i|$ chemins simples. Chaque sommet v_i est le début d'un de ces chemins simples. Les sommets successifs de ces chemins sont les plus différents possibles les uns des autres. Nous ne souhaitons pas avoir uniquement des contours de même orientation.

Nous utilisons ensuite le noyau sur sacs de Wallraven [WCG03].

$$\begin{aligned} K_{avgMax}(B_i, B_j) &= \frac{1}{|B_i|} \sum_{j=1}^{|B_j|} \max_i \; k(h_{v_i}, h_{v_j}) \\ &+ \frac{1}{|B_j|} \sum_{i=1}^{|B_i|} \max_j \; k(h_{v_i}, h_{v_j}) \end{aligned} \tag{5.1}$$

avec

$$
\begin{aligned}
k(h_{v_i}, h_{v_j}) &= K_C(h_{v_i}, h_{v_j}) \\
&= K_v(v_i, v_j) + \sum_{n=1}^{|h|} Se_n O_{n,n-1} K_e(e_n, e'_n)\ K_v(v_n, v'_n).
\end{aligned}
$$

Sur la figure (Fig.(5.12)), nous notons que la précision moyenne est dégradée avec l'utilisation du noyau sur sacs (eq.(5.1)). Les résultats sont moins bons car la sélection des chemins est trop contrainte et empêche de trouver l'appariement optimal entre tous les chemins des graphes.

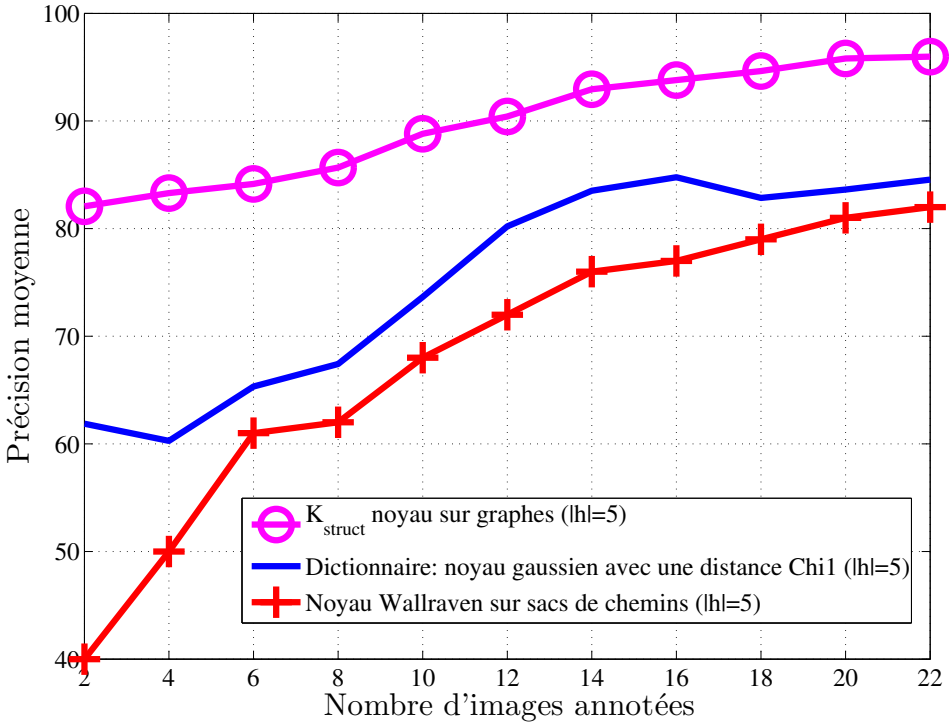


FIGURE 5.12 – Comparaison des résultats dictionnaire, noyau sur sac de contours et noyau sur graphes K_{struct}

5.2.4 Résultats de session de recherche

Dans le but d'illustrer davantage le noyau sur graphe K_{struct}, nous présentons des sessions de recherche sur différents objets. Un utilisateur recherche une

catégorie d'images à l'aide de l'interface graphique RETIN. Au fur et à mesure des itérations, le système classe la base puis l'utilisateur annote une sélection d'images pertinentes, puis réitère le classement (Une description du système est détaillée dans le chapitre 4 (section résultat 4.2.1.2)).

Nous testons sur la base de 220 images avec 70 fenêtres et 150 fausses alarmes (Fig.(5.4)). Sur la figure (Fig.(5.13)) le classement pour une requête de fenêtre est représenté (image située en haut à gauche avec un carré vert). Sur la figure (Fig.(5.13), de gauche à droite, et de haut en bas, nous avons les 20 premières images du classement. Sur la figure (Fig.(5.13)), il s'agit des 20 suivantes.

A ce stade, nous pouvons observer que le système a sélectionné des images parmi les plus pertinentes (5 images en bas de la figure (Fig.(5.13)). Nous avons annoté ces images et mis à jour le classement (Fig.(5.13)). Dans ce classement, nous remarquons que le système propose une image négative dans la sélection active (Fig.(5.15)). Nous l'annotons négativement (Fig.(5.16)) et mettons à jour le système. Le nouveau classement est montré sur la figure (Fig.(5.17)).

Afin d'évaluer, cette section de recherche nous avons regardé le classement des 70 premières images (la base ayant 70 fenêtres). Dans le tableau (Tab.(5.3)), nous pouvons apprécier que le classement de tête s'améliore d'itération en itération.

Itération	Annotation	TOP 70
0	1 positif	23 négatifs
1	5 positifs	19 négatifs
2	5 positifs et 1 négatifs	14 négatifs
3	5 positifs et 4 négatifs	11 négatifs

TABLE 5.3 – **Session de recherche après 3 itérations.** L'utilisateur annote positivement ou négativement des exemples. Dans les 70 premières images du classement, nous regardons le nombre de négatifs.

FIGURE 5.13 – Les 20 premières images du classement pour une requête en haut à gauche

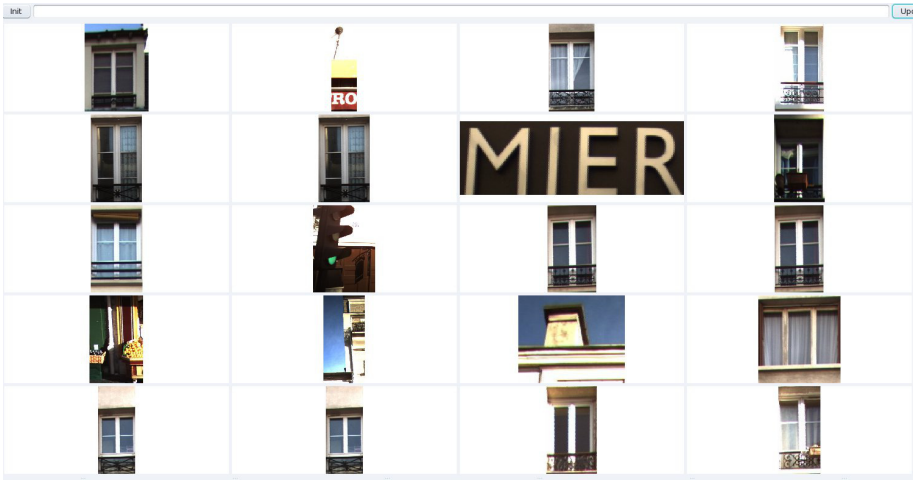


FIGURE 5.14 – Les 20 images suivantes (20 à 40) du classement

FIGURE 5.15 – Classement après une itération et 5 images annotées

FIGURE 5.16 – Annotation d'une image négative avant mise à jour du classement

FIGURE 5.17 – Classement après 2 itérations et 6 images annotées dont une négative

Conclusions et perspectives

Conclusions

Nos travaux ont porté sur l'extraction de fenêtres et l'étude d'un modèle de mise en correspondance des contours. Ces travaux peuvent-être divisés en deux parties distinctes.

Dans une première partie, nous avons proposé d'extraire des hypothèses de fenêtre à l'aide d'un algorithme automatisé qui s'appuie sur les propriétés des fenêtres. Nous avons apporté deux contributions à l'algorithme inspiré d'un algorithme de l'état de l'art de Lee et Nevatia. La première contribution est la recherche des étages puis des fenêtres par étages. La deuxième contribution est l'automatisation de la recherche du meilleur facteur d'échelle. Ainsi nous cherchons le meilleur paramètre de lissage et dérivation qui permet de garder les gradients principaux des fenêtres et d'écarter le bruit. Suite à une évaluation des deux algorithmes, notre algorithme permet de détecter plus de fenêtres avec moins de faux-positifs.

Dans une deuxième partie, nous avons proposé un modèle de mise en correspondance de chemins de contours afin d'exploiter les contours pour effectuer la recherche d'objets dans une base d'images mais aussi pour localiser ces objets dans l'image. La principale difficulté est d'obtenir le regroupement des contours appartenant à l'objet recherché. Le modèle proposé est d'utiliser des noyaux sur graphes (noyaux sur sacs de chemins) à l'aide d'un nouveau noyau sur chemins. Nous avons proposé des similarités sur chemins robustes aux changements d'échelle qui prennent en compte l'orientation des contours et la structure de l'objet. Lors de la classification d'hypothèses de fenêtres, la similarité sur chemins à l'aide des contours était suffisante, mais lors de la détection d'objet dans les façades, il a été nécessaire d'adjoindre une information issue de la radiométrie des images pour

renforcer la similarité des ensembles de contours. Nous avons alors proposé des similarités mixant informations contours et régions délimitées par ces contours. L'apport de l'information région a permis de renforcer les bons appariements et de réduire l'influence des appariements de bruit.

Le noyau doit donner de bonnes performances en un temps suffisamment rapide. Par rapport, au noyau de l'état de l'art, nous avons choisi de ne pas explorer tous les chemins possibles mais de trouver parmi un ensemble restreint les meilleurs appariements. Les performances sont bonnes mais malgré nos propositions pour exploiter l'arbre de recherche, la complexité de calcul reste encore un problème pour pouvoir faire une recherche dans de plus larges bases et des graphes plus grands.

Par ailleurs, pour régler le problème de la complexité de calcul, nous avons proposé un nouveau noyau sur des graphes basés sur un dictionnaire de chemins et le calcul d'un noyau incrémental. Ce dictionnaire est construit dynamiquement au cours de la session de recherche, les mots (chemins de contours) étant pris à partir des images pertinentes. Si les résultats sont légèrement moins précis que la comparaison sur noyau sur graphes, le processus est cependant dix fois plus rapide !

Enfin, l'algorithme de "séparation et évaluation" employé avec ces nouveaux noyaux permet de trouver des solutions exactes ou approchées au problème de l'appariement de graphes avec des graphes ou des sous-graphes d'autres images. Intégrées dans des classifieurs de type SVM ou kppv, les similarités proposées permettent de retrouver (et de classer) des images comme contenant un objet particulier décrit par l'exemple.

Perspectives

Le travail réalisé dans le cadre de cette thèse offre plusieurs perspectives d'évolution.

A l'origine de cette thèse, nous souhaitions proposer un noyau le plus généraliste possible. Cependant la complexité de calcul a fait que nous nous sommes concentrés plus sur la structure des graphes et sur la similarité que sur les labels des sommets et des arêtes. En effet les attributs des sommets et des arêtes possèdent peu d'informations (orientation du contour et position). Nous pourrions réfléchir à d'autres attributs qui permettent de décrire des contours courbés et

plus uniquement des segments de contours. Apporter plus d'information au niveau des contours pourrait probablement mieux décrire des objets plus complexes comme une voiture par exemple. En parallèle d'une analyse plus poussée sur les attributs des sommets, les poids (orientation et échelle) devront être adaptés pour représenter les caractéristiques contours des objets plus complexes. Par exemple le poids sur l'orientation est intéressant pour un objet comme la fenêtre où la séquence des contours principaux est un contour horizontal puis vertical, mais sur des objets plus complexes ce poids est-il toujours aussi pertinent ?

La principale amélioration concerne la complexité de calcul. Nous nous sommes concentrés dans cette thèse plutôt sur la caractérisation des contours en graphes et sur les noyaux possibles. En ce qui concerne la complexité de calcul proprement dite, nous avons adapté des algorithmes d'exploration d'arbre de recherche de l'état de l'art et une approche innovante basée sur la construction incrémentale d'un dictionnaire. Il serait intéressant de voir les résultats avec un code parallélisé sur carte graphique (GPU) par exemple. L'accélération des temps de calcul permettrait de tester d'autres objets qui nécessitent de plus grands graphes de contours pour les représenter.

Le dernier point abordé lors de cette thèse fut l'implémentation de dictionnaire de chemins de contours afin de réduire l'ensemble de recherche des meilleurs appariements et ainsi réduire le temps de calcul. Les résultats étaient moins précis mais le temps de calcul était réduit. Lors de l'ajout d'un nouveau chemin dans le dictionnaire, nous avons fait le choix de ne pas vérifier si un chemin similaire existait déjà dans le dictionnaire. Une perspective intéressante est de pouvoir éliminer les chemins du dictionnaire très proches en termes de similarité. Il faudrait donc éliminer les bouts de chemin qui reviennent régulièrement de même qu'en recherche d'information textuelle où on élimine en pré-traitement les mots ne portant pas beaucoup d'information mais très présents dans les documents ("le", "de", "et",...). La question sur la façon de caractériser cette distance reste ouverte.

Bibliographie

[ABC01] T. Amghar, D. Battistelli, and T. Charnois. Représenter le temps en langue dans le formalisme des graphes conceptuels : une approche basée sur les schèmes sémantico-cognitifs. 2001.

[AD04] O. Alegre and F. Dellaert. A probabilistic approach to the semantic interpretation of building facades. In *In Int. Workshop on Vision Techniques Applied*, pages 1–12, 2004.

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, 1983.

[ASJ+07] Haider Ali, Christin Seifert, Nitin Jindal, Lucas Paletta, and Gerhard Paar. Window detection in facades. In *ICIAP 07 : Proceedings of the 14th International Conference on Image Analysis and Processing*, pages 837–842, Washington, DC, USA, 2007. IEEE Computer Society.

[Aur91] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3) :345–405, 1991.

[BDBV03] S. Berretti, A. Del Bimbo, and E. Vicario. Weighted walkthroughs between extended entities for retrieval by spatial arrangement. *IEEE Transactions on Multimedia*, 5(1) :52–70, 2003.

[Ber58] C. Berge. La Theorie des Graphes. *Paris, France*, 1958.

[BJ88] I. Biederman and G. Ju. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology*, 20(1) :38–64, 1988.

[Blo99] I. Bloch. Fuzzy relative position between objects in image processing : a morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7) :657, 1999.

[BMP00] S. Belongie, J. Malik, and J. Puzicha. Shape context : A new descriptor for shape matching and object recognition. In *In NIPS*. Citeseer, 2000.

[Bog74] J. Bognár. *Indefinite inner product spaces*. Springer, 1974.

[BOS+05] K.M. Borgwardt, C.S. Ong, S. Schönauer, S.V.N. Vishwanathan, A.J. Smola, and H.P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics-Oxford*, 21(1) :47–56, 2005.

[BR06] C. Brenner and N. Ripperda. Extraction of facades using rjmcmc and constraint equations. In *PCV06*, 2006.

[BR07] H. Bunke and K. Riesen. A family of novel graph kernels for structural pattern recognition. In *CIARP*, volume 4756, pages 20–31, 2007.

[BS98] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3-4) :255–259, 1998.

[CCB] W.L. Chan, H. Choi, and R. Baraniuk. Quaternion wavelets for image analysis and processing. In *ICIP*, volume 5, pages 3057–3060.

[CFSV01] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159. Citeseer, 2001.

[CFSV04] D. Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3) :265–298, 2004.

[CH01] M. Carcassoni and E. R. Hancock. Weighted graph-matching using modal clusters. In *IAPR-TC15 Workshop Graph-Based Representations in Pattern Recognition*, pages 260–269, 2001.

[CHP97] M. Cord, F. Huet, and S. Philipp. Optimal adjusting of edge detectors to extract close contours. In *Scandinavian conference on image analysis*, pages 627–633, 1997.

[Cla97] J. Clausen. Branch and bound algorithms-principles and examples. *Parallel Computing in Optimization*, pages 239–267, 1997.

[CV99] R. Casati and A.C. Varzi. *Parts and places : the structures of spatial representation*. The MIT Press, 1999.

[CW04] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *International Journal on Machine Learning Research*, 5 :913–939, 2004.

[DB09] F.X. Dupé and L. Brun. Edition within a graph kernel framework for shape recognition. *Graph-Based Representations in Pattern Recognition*, pages 11–20, 2009.

[DBKP09] Olivier Duchenne, Francis Bach, Inso Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1980–1987, November 2009.

[DTC04] A. R. Dick, P. H. S. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *Int. J. Comput. Vision*, 60(2) :111–134, 2004.

[EC04] J. Eichhorn and O. Chapelle. Object categorization with svm : kernels for local features. Technical report, Max Planck Institute, 2004.

[EM95] M. Egenhofer and D. Mark. Naive geography. *Spatial Information Theory A Theoretical Basis for GIS*, pages 1–15, 1995.

[EWH09] David Emms, Richard C. Wilson, and Edwin R. Hancock. Graph matching using the interference of continuous-time quantum walks. *Pattern Recognition*, 42(5) :985–1002, 2009.

[FFJS08] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1) :36–51, 2008.

[FGB09] S. Fischer, K. Gilomen, and H. Bunke. Identification of diatoms by grid graph matching. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 335–370, 2009.

[FGK95] A. Filatov, A. Gitis, and I. Kil. Graph-based handwritten digit string recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2)-Volume 2*, page 845. IEEE Computer Society, 1995.

[FJS09] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *IJCV à paraître*, 2009.

[Fre92] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial intelligence*, 54(1-2) :199–227, 1992.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *In Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[G̈94] R.H. Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4) :357–399, 1994.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1) :49–58, 2003.

[GC06] P.H. Gosselin and M. Cord. Precision-oriented active selection for interactive image retrieval. In *IEEE International Conference on*

Image Processing, pages 3197–3200, Atlanta, GA, USA, October 2006.

[GCPF07a] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Kernel on bags of fuzzy regions for fast object retrieval. In *IEEE International Conference on Image Processing*, volume 1, pages 177–180, San Antonio, Texas, USA, September 2007.

[GCPF07b] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Kernels on bags for multi-object database retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, page 231. ACM, 2007.

[GFW03] T. Gartner, P. Flach, and S. Wrobel. On graph kernels : Hardness results and efficient alternatives. *Lecture notes in computer science*, pages 129–143, 2003.

[GH04] I. Gondra and D.R. Heisterkamp. Learning in region-based image retrieval with generalized support vector machines. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, volume 27, pages 149–149, june 2004.

[GPS11] P.H Gosselin, F. Precioso, and Philipp-Foliguet S. Incremental kernel learning for active image retrieval without global dictionaries. *Pattern Recognition*, 2011.

[Haa05] Bernard Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4) :482–492, 2005.

[Hau99] David Haussler. Convolution kernels on discrete structures. Technical report, Institute of California at Santa Cruz, 1999.

[HC01] S. Hazarika and A. Cohn. Qualitative spatio-temporal continuity. *Spatial Information Theory*, pages 92–107, 2001.

[HM08] Jorge Hernàndez and Beatriz Marcotegui. Ultimate opening segmentation with shape constraints. In *Visualization, Imaging, and Image Processing (VIIP 2008)*, Palma de Mallorca, Spain, 2008.

[HP08] B. Haasdonk and E. Pekalska. Indefinite kernel fisher discriminant. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 1–4, Tampa, USA, Dec. 2008.

[IM06] T. Ishikawa and D.R. Montello. Spatial knowledge acquisition from direct experience in the environment : Individual differences in the development of metric knowledge and the integration of separately learned places. *Cognitive Psychology*, 52(2) :93–129, 2006.

[Joh02] Björn Johansson. *Computer Vision Using Rich Features - Geometry and Systems.* PhD thesis, Lund University, 2002.

[KC02] S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. In *IAPR International Workshops on SSPR and SPR*, pages 133–142, 2002.

[KJ03] R. Kondor and T. Jebara. A kernel between sets of vectors. In *International Conference on Machine Learning (ICML)*, 2003.

[KKM93] R. Krishnapuram, J.M. Keller, and Y. Ma. Quantitative analysis of properties and spatial relations of fuzzyimage regions. *IEEE Transactions on Fuzzy Systems*, 1(3) :222–233, 1993.

[KT04] H. Kashima and Y. Tsuboi. Kernel-based discriminative learning algorithms for labeling sequences, trees and graphs. In *International Conference on Machine Learning (ICML)*, page 58, Banff, Alberta, Canada, 2004.

[LDR+07] A.K. Lautenschütz, C. Davies, M. Raubal, A. Schwering, and E. Pederson. The influence of scale, context and spatial preposition in linguistic topology. *Spatial Cognition V Reasoning, Action, Interaction*, pages 439–452, 2007.

[LH05] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, pages 1482–1489, October 2005.

[LKP03] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM-Symposium*, pages 297–304, 2003.

[LN04] Sung Chun Lee and Ramakant Nevatia. Extraction and integration of window in a 3d building model from ground view image. In *CVPR (2)*, pages 113–120, 2004.

[LPFG08] J. Lebrun, S. Philipp-Foliguet, and P.-H. Gosselin. Image retrieval with graph kernel on regions. In *19th ICPR International Conference on Pattern Recognition*. IEEE, USF, IEEE, dec 2008.

[LSB06] B. Le Saux and H. Bunke. Combining SVM and Graph Matching in a Bayesian Multiple Classifier System for Image Content Recognition. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 696–704, 2006.

[Lyu05] S. Lyu. Mercer kernels for object recognition with local features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 223–229, San Diego, CA, 2005.

[MB98] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 493–504, 1998.

[MBPS09] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*, pages 689–696, 2009.

[Mes95] B. Messmer. *Efficient Graph Matching Algorithm for Preprocessed Model Graphs*. PhD thesis, Institut fur Informatik und Angewandte Mathematik, University of Bern, 1995.

[MNJ08] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30 :1632–1646., september 2008.

[MR07] H. Mayer and S. Reznik. Building facade interpretation from uncalibrated wide-baseline image sequences. *PandRS*, 61(6) :371–380, February 2007.

[MV09] P. Mahé and J.-P. Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1) :3–35, 2009.

[MVNT+10] Oussama Moslah, Daniel Voinea, Rodrigue Nkoutche-Tedjong, Alejandro Sanchez-Guinea, Yassine Khial, Vincent Le Moigne, Serge Couvet, Sylvie Philipp-Foliguet, and Peter Wonka. A model-based facade reconstruction approach using shape grammars. In *ACM Transactions on Graphics*, 2010.

[MZM02] J. Malki, E.H. Zahzah, and L. Mascarilla. Indexation et recherche d'image fondées sur les relations spatiales entre objets= Indexing and retrieval based on spatial relationships between objects. *TS. Traitement du signal*, 19(4) :235–251, 2002.

[MZWG07] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. 26(3), 2007.

[OMCS04] C.S. Ong, X. Mary, S. Canu, and A.J. Smola. Learning with non-positive kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 81. ACM, 2004.

[OPZ06] A. Opelt, A. Pinz, and A Zisserman. A boundary-fragment-model for object detection. In *ECCV*, pages 575–588, 2006.

[PFG06] S. Philipp-Foliguet and J. Gony. FReBIR : Fuzzy regions-based image retrieval. In *Information Processing and Management of Uncertainty (IPMU)*, Paris, France, July 2006.

[PFGG09] S. Philipp-Foliguet, J. Gony, and P.H. Gosselin. FReBIR : An image retrieval system based on fuzzy region matching. *Computer Vision and Image Understanding*, 113(6) :693–707, 2009.

[PFVL05] S. Philipp-Foliguet, M.B. Vieira, and M. Lekkat. Recherche d'images par appariement d'ensembles de régions floues. *Information, Interaction, Intelligence*, 5(2) :9–40, 2005.

[PH09] E. Pekalska and B. Haasdonk. Kernel Discriminant Analysis for Positive Definite and Indefinite Kernels. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 31(6) :1017, 2009.

[Sak84] M. Sakarovitch. *Optimisation combinatoire :. Graphes et programmation linéaire*. Hermann, 1984.

[SB03] Konrad Schindler and Joachim Bauer. A model-based method for building reconstruction. In *HLK '03 : Proceedings of the First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, Washington, DC, USA, 2003. IEEE Computer Society.

[SB09] N. Shervashidze and K. M. Borgwardt. Fast subtree kernel on graphs. In *Neural Information Processing Systems (NIPS)*, volume 22, pages 1660–1668, 2009.

[SBC05] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *ICCV*, pages 503–510, 2005.

[SBM+06] A. Shokoufandeh, L. Bretzner, D. Macrini, M. Fatih Demirci, C. Jönsson, and S. Dickinson. The representation and matching of categorical shape. *Computer Vision and Image Understanding*, 103(2) :139, 2006.

[SBSS00] A.J. Smola, P. Barlett, B. Scholkopf, and C. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, 2000.

[SBV01] K. Shearer, H. Bunke, and S. Venkatesh. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *Pattern Recognition*, 34(5) :1075–1091, 2001.

[SC92] J. Shen and S. Castan. An optimal linear operator for step edge detection. *CVGIP : Graphical Models and Image Processing*, 54(2) :112–133, 1992.

[SGRB05] Frederic Suard, Vincent Guigue, Alain Rakotomamonjy, and Abdelaziz Bensrhair. Pedestrian detection using stereo-vision and graph kernels. In *Intelligent Vehicles Symposium*, pages 267–272, Las Vegas, Nevada, June 2005.

[SS02] B. Sch
"olkopf and A.J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. the MIT Press, 2002.

[STC04] J. Shawe-Taylor and N. Cristianini. *Kernel methods for Pattern Analysis*. Cambridge University Press, ISBN 0-521-81397-2, 2004.

[Sua06] F. Suard. *Méthodes à noyaux pour la détection de piétons.* PhD thesis, Institut National des Sciences Appliquées de Rouen, 2006.

[SVP+09] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 488–495, 2009.

[SY98] PN Suganthan and H. Yan. Recognition of handprinted Chinese characters by constrained graph matching. *Image and Vision Computing*, 16(3) :191–201, 1998.

[TK01] S Tong and D Koller. Support vector machine active learning with application to text classification. *Journal of Machine Learning Research*, pages 2 :45–66, November 2001.

[Tor04] A. Torsello. *Matching hierarchical structures for shape recognition.* PhD thesis, University of York, 2004.

[TSGa] TSG-20 : Tourist Sights Graz Image Database. http ://dib.joanneum.at/cape/TSG-20/.

[TSGb] TSG-60 : Tourist Sights Graz Image Database. http ://dib.joanneum.at/cape/TSG-60/.

[Ull76] J. R. Ullman. An algorithm for subgraph isomorphism. In *J. Assoc. Comput.*, pages 31–42, March 1976.

[Ume88] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intellenge*, 10 :695–703, 1988.

[Vap98] V.N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, New York, 1998.

[VBKS08] S.V.N. Vishwanathan, K.M. Borgwardt, I.R. Kondor, and N.N. Schraudolph. Graph kernels. *Journal of Machine Learning Research*, 9 :1–37, 2008.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.

[Vor08] G. Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik (Crelle's Journal)*, 1908(133) :97–102, 1908.

[VSKB09] S. V. N. Vishwanathan, Nicol N. Schraudolph, Imre Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 10, 2009.

[WCG03] C. Wallraven, B. Caputo, and A.B.A. Graf. Recognition with local features : the kernel recipe. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 257–264, 2003.

[WH03] R. C. Wilson and E. R. Hancock. Pattern spaces from graph polynomials. In *12th International Conference On Image Analysis And Processing*, pages 480–485, 2003.

[WZ02] T. Werner and A. Zisserman. New techniques for automated architecture reconstruction from photographs. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 2, pages 541–555. sp, 2002.

[XK01] L. Xu and I. King. A pca approach for fast retrieval of structural patterns in attributed graphs. *IEEE Transactions on Systems, Man and Cybernetics*, 31 :812–817, 2001.

[ZuB] ZuBuD :Zurich Building Image Database. http ://www.vision.ee.ethz.ch/showroom/zubud/index.en.html.

[ZW08] P. Zhu and RC Wilson. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9) :2833–2841, 2008.

Résumé

Cette dernière décennie, la modélisation des villes 3D est devenue l'un des enjeux de la recherche multimédia et un axe important en reconnaissance d'objets. Dans cette thèse nous nous sommes intéressés à localiser différentes primitives, plus particulièrement les fenêtres, dans les façades de Paris. Dans un premier temps, nous présentons une analyse des façades et des différentes propriétés des fenêtres. Nous en déduisons et proposons ensuite un algorithme capable d'extraire automatiquement des hypothèses de fenêtres. Dans une deuxième partie, nous abordons l'extraction et la reconnaissance des primitives à l'aide d'appariement de graphes de contours. En effet une image de contours est lisible par l'oeil humain qui effectue un groupement perceptuel et distingue les entités présentes dans la scène. C'est ce mécanisme que nous avons cherché à reproduire. L'image est représentée sous la forme d'un graphe d'adjacence de segments de contours, valué par des informations d'orientation et de proximité des segments de contours. Pour la mise en correspondance inexacte des graphes, nous proposons plusieurs variantes d'une nouvelle similarité basée sur des ensembles de chemins tracés sur les graphes, capables d'effectuer les groupements des contours et robustes aux changements d'échelle. La similarité entre chemins prend en compte la similarité des ensembles de segments de contours et la similarité des régions définies par ces chemins. La sélection des images d'une base contenant un objet particulier s'effectue à l'aide d'un classifieur SVM ou kppv. La localisation des objets dans l'image utilise un système de vote à partir des chemins sélectionnés par l'algorithme d'appariement.

Abstract

This last decade, modeling of 3D city became one of the challenges of multimedia search and an important focus in object recognition. In this thesis we are interested to locate various primitive, especially the windows, in the facades of Paris. At first, we present an analysis of the facades and windows properties. Then we propose an algorithm able to extract automatically window candidates. In a second part, we discuss about extraction and recognition primitives using graph matching of contours. Indeed an image of contours is readable by the human eye, which uses perceptual grouping and makes distinction between entities present in the scene. It is this mechanism that we have tried to replicate. The image is represented as a graph of adjacency of segments of contours, valued by information orientation and proximity to edge segments. For the inexact matching of graphs, we propose several variants of a new similarity based on sets of paths, able to group several contours and robust to scale changes. The similarity between paths takes into account the similarity of sets of segments of contours and the similarity of the regions defined by these paths. The selection of images from a database containing a particular object is done using a KNN or SVM classifier.