UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DÉTECTION DE RELATIONS SPATIALES ENTRE LES OBJETS PRÉSENTS DANS UNE IMAGE

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

DAVID DURAND

MAI 2006

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Ce mémoire intitulé :

DÉTECTION DE RELATIONS SPATIALES ENTRE LES OBJETS PRÉSENTS DANS UNE IMAGE

présenté par
David Durand

pour l'obtention du grade de maître ès science (M.Sc.)

a été évalué par un jury composé des personnes suivantes :

Dr. Rokia Missaoui ..... Directrice de recherche
Dr. Marek Zaremba ..... President du jury
Dr. Nadia Baaziz ..... membre du jury

Mémoire accepté le : 23 mai 2006

J'adresse mes remerciements à ma famille, pour m'avoir encouragé et soutenu tout au long de mon séjour au Québec, et plus encore lors de mes passages en France

à Angela, pour sa patience face à mes nombreuses heures passées devant l'ordinateur,

à Rokia Missaoui pour son aide précieuse à l'élaboration de ce mémoire,
et à Sarifuddin pour ses conseils dans la réalisation.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations, sigles et acronymes

BD Bases de Données

RIC Recherche d'Image selon le Contenu

RICS Recherche d'Image selon le Contenu par Similarité

DRS Détection de Relations Spatiales

SGBD Système de Gestion de Bases de Données

Résumé

La recherche d'image selon le contenu (RIC) fait appel à diverses caractéristiques visuelles dont la couleur, la forme et la texture ainsi qu'aux métadonnées (ex. annotations d'images) en vue de récupérer des images similaires à une image donnée ou répondant à des critères spécifiques. Afin de raffiner le processus de RIC, nous avons conçu et implanté des procédures de détection des relations spatiales entre les objets présents dans une image.

La revue de la littérature dans le domaine de la RIC fait référence à huit primitives de relations spatiales et fait état de plusieurs travaux sur l'exploitation de ces types de relations pour une recherche plus efficace dans une collection d'images. Cependant, il n'y a eu que peu d'études sur des algorithmes et des techniques d'identification de ces relations dans les images. Notre travail vise justement à combler cette lacune en mettant au point de nouvelles procédures de découverte de relations spatiales. La validation et la justification de notre recherche implique le processus suivant : (i) identification des objets contenus dans une image, (ii) détection des relations spatiales entre les objets, et (iii) analyse de l'efficacité des performances de la méthode de détection de relations spatiales sur des images prototypes.

Abstract

Content-based image retrieval (CBIR) relies on different types of visual characteristics such as color, shape, texture and metadata to retrieve images that are either similar to a given one, or verifying some predefined criteria. In this work, we have designed and implemented procedures to detect spatial relationships between objects in digital images so that such links can be further exploited to refine the CBIR process.

The literature in CBIR makes generally reference to eight primitives for spatial relationships, and contains several studies on the way to use, rather than automatically identify, those relationships for a more efficient retrieval in image collections. Our contribution aims at filling this gap by developing new procedures to discover spatial relationships. The validation and justification of our approach is done as follows : (i) identification of the objects contained in an image, (ii) detection of the spatial relationships between those objects, and (iii) performance analysis of the detection process on sample images.

Introduction

Ce projet de recherche se situe dans le cadre de l'analyse d'image et plus particulièrement au niveau de la description automatique des images. Ce domaine de l'analyse d'image a été introduit pour répondre à un besoin lié aux bases de données (BD) multimédia. Les énormes quantités d'images stockées dans certaines bases de données ont amené les utilisateurs de telles BD à vouloir mettre en place un système de recherche d'image basée sur le contenu visuel de ces images. Dans cette optique ont été créés des algorithmes de segmentation d'image selon les couleurs ou les textures, de détection des contours ou encore de reconnaissance de formes.

Plus récemment, le besoin d'étudier les relations spatiales entre les objets dans une image s'est fait sentir non seulement pour les BD multimédia et les systèmes d'information géographique, mais aussi dans d'autres disciplines parmi lesquelles l'intelligence artificielle ou les sciences cognitives, ainsi que dans des domaines plus éloignés comme la médecine ou la cartographie. Des études ont donc été menées sur le sujet et des modèles de relations spatiales ont été établis. Ces modèles sont toutefois restés au stade de modèles mathématiques non exploitables directement dans un programme informatique. Nous nous sommes donc proposés de convertir ces modèles mathématiques de relations spatiales en modèles informatiques et de les implanter au sein d'algorithmes de détection de relations spatiales dans une application d'analyse d'image.

Le processus de détection des relations spatiales (DRS) entre les objets présents dans une image consiste à nommer des liens présents entre les objets isolés dans l'image. Le problème ici est l'émulation du jugement humain : un programme informatique est incapable d'affirmer par exemple qu'un arbre se trouve **devant** une maison sur une photographie car cela pose un problème au niveau de la définition formelle de la notion

exprimée par le terme « devant ». Les premières études sur le sujet ont permis de définir formellement ces relations et de dégager huit relations *primitives* à partir desquelles peuvent se déduire les autres.

Les résultats de ces études nous ont servi de point de départ pour développer des méthodes permettant de détecter les relations spatiales dans une image sachant que l'objectif final du projet est la détection exacte des huit relations primitives.

Le travail abordé dans ce projet de recherche fait appel aux notions classiques de traitement d'image telles que la décomposition d'une image selon des plages de couleur, l'extraction des contours et le suivi de ces contours.

Ce document est organisé de la façon suivante. En premier lieu seront présentées quelques bases théoriques de la DRS, ensuite nous continuerons avec l'état de l'art en présentant les algorithmes existants pour le traitement de la DRS et en examinant les lacunes qu'ils possèdent et qui justifient notre démarche. Le chapitre suivant présentera le raisonnement que nous avons suivi et les méthodes que nous avons mises au point pour la DRS. L'implémentation de ces modèles théoriques sous forme d'algorithmes sera décrite dans le chapitre suivant, tandis que les deux derniers chapitres présenteront les expérimentations que nous avons mises en place pour valider notre système, les résultats que nous avons obtenus, et les conclusions que nous en avons tirés.

Chapitre 1

La détection de relations spatiales

La détection de relations spatiales entre les objets présents dans une image est une des composantes de la description automatique d'image. Nous allons donc voir ici en quoi consiste la description automatique d'image, quelles sont les différents types d'informations collectés pour établir une telle description et par quels moyens elles sont collectées, et finalement où se situe la DRS dans le procédé et quels sont ses relations avec les autres analyses effectuées durant ce processus.

1.1 Contexte

1.1.1 La description automatique d'image

La description automatique d'image est un des domaines de recherche les plus en vogue actuellement dans le traitement d'image. Le problème est apparu en particulier dans les bases de données multimédia. Celles-ci ayant atteint des tailles considérables, leurs utilisateurs ont commencé à être confrontés au problème de recherche d'image au sein de ces BD. La question posée est la suivante : « comment retrouver dans une BD de plusieurs millions d'images celles que l'on recherche précisément ? » La réponse très simple à cette question est : « en spécifiant précisément les éléments qui doivent être présents sur cette image ». Le moteur de recherche n'aura plus qu'à comparer ces caractéristiques avec celles des images contenues dans la base. Le problème c'est que ceci requiert d'avoir au préalable rentré les caractéristiques de chaque image dans la

base de donnée. Et si c'est faisable manuellement pour les petites BD, c'est par contre inconcevable pour celles qui dépassent les milliers d'enregistrements.

Plusieurs équipes ont donc commencé à s'intéresser au problème de description automatique d'image. Des systèmes d'annotation automatique d'image ont été élaborés [16] pour permettre des recherches par mots-clés, mais ces procédés sont en réalité situés en aval des processus d'analyse d'image et n'extraient pas réellement directement l'information de l'image. C'est plutôt le deuxième type de méthodes qui nous intéresse ici : les méthodes d'extraction de contenu par reconnaissance de forme, de couleur ou de texture. L'idée est de segmenter l'image en objets [12], pour pouvoir décrire séparément chacun de ces objets, grâce à des bases de connaissances et à des algorithmes de reconnaissance d'objet, puis de décrire de quelle manière ils interagissent dans l'image. C'est ici qu'intervient la DRS.

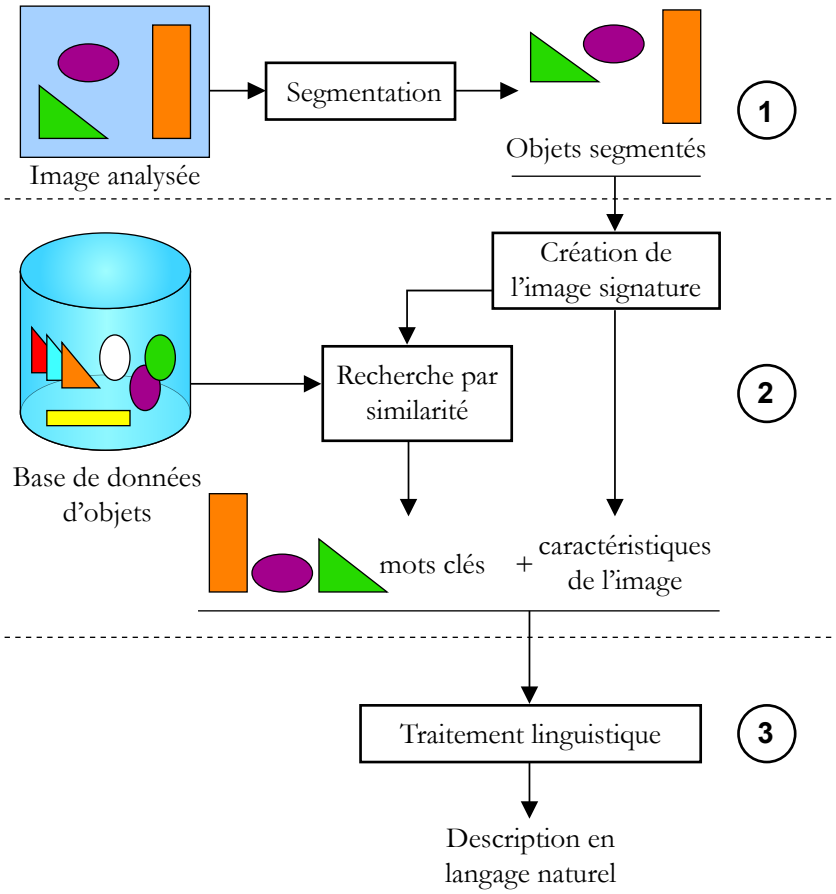


FIG. 1.1 – Étapes de l'annotation automatique d'image

La figure 1.1 montre les trois étapes du processus d'annotation automatique d'images de l'algorithme de Hède, Moëllic, Bourgeoys, Joint et Thomas [16]. La DRS, qui n'apparaît pas dans cette figure, se situe dans la création de l'image signature. Elle permettrait ici d'obtenir des informations sur les relations existant entre les objets en plus des informations sur les objets eux-mêmes (nature, couleur etc...) déjà collectées par leur système.

1.1.2 La détection de relations spatiales

La détection de relations spatiales dans une image est une fonctionnalité très demandée à l'heure actuelle dans des domaines divers et variés. Parmi ces domaines, on peut relever non seulement les systèmes de gestion de bases de données, comme indiqué dans la section précédente, mais aussi l'imagerie informatique, la conception graphique assistée par ordinateur, la robotique, les systèmes d'information géographique, le traitement d'image, la géométrie informatique ou encore la reconnaissance de modèles [25]. Nous avons toutefois choisi quant à nous de concentrer nos efforts sur les applications dans les systèmes de gestion de bases de données (SGBD) traitant des bases d'images. Plus particulièrement, les bases d'images atteignent à présent des tailles considérables – le système d'observation par satellite de la NASA génère par exemple un tera-octet de données par jour de fonctionnement –, il devient donc nécessaire de mettre au point des techniques de recherche d'images adaptées, et l'orientation actuelle est à la recherche d'image basée sur le contenu. La recherche d'image basée sur le contenu (RIC) se présente sous deux formes différentes :

- Une recherche basée sur des critères définis par l'utilisateur : c'est à ce dernier de spécifier les critères qu'il veut voir apparaître dans les images, en indiquant par exemple qu'il souhaite voir apparaître une certaine quantité de bleu et une certaine quantité de rouge, qu'il souhaite aussi avoir des formes plutôt rondes qu'angulaires, et enfin en précisant à l'aide de mots-clés le thème de l'image, par exemple, un paysage, ou des constructions etc...
- Une recherche prenant une image donnée comme référence : ici, l'utilisateur donne simplement au programme une image de référence, et c'est le programme qui se charge d'extraire les informations de l'image, de manière transparente vis-à-vis de l'utilisateur, et va ensuite chercher dans la BD les images ayant des

caractéristiques similaires. Le problème, comme nous le verrons par la suite, étant de définir le terme « similaire » dans ce cas [26, 22, 21].

L'approche la plus utilisée dans la littérature est la seconde, soit la recherche d'image par similarité avec une image référence. C'est cette approche qui va nous intéresser plus particulièrement.

Dans ces deux cas toutefois, la recherche peut prendre en compte (simultanément ou séparément) des critères de couleur, de texture, de formes, de volume ou de mouvement ainsi que des contraintes sémantiques ou spatiales [10]. Nous allons voir dans le chapitre 3 quels critères seront utiles à notre démarche, et comment ils seront utilisés.

1.2 La recherche d'images basée sur le contenu

Cette partie va permettre au lecteur de mieux comprendre où se place le problème que nous allons étudier dans le processus de traitement d'une image pour en extraire des caractéristiques visuelles.

La recherche d'image basée sur le contenu consiste non seulement à compléter les requêtes textuelles imparfaites mais aussi à s'inspirer du système visuel de l'être humain pour dégager des analyses d'image proches des requêtes des utilisateurs [28]. Le système visuel de l'être humain, contrairement à un ordinateur, ne perçoit pas une image comme une juxtaposition de pixels, mais plutôt comme un ensemble d'objets qu'il est capable de distinguer au sein d'une grande collection d'images [26]. Il est de plus capable de reconnaître les interactions présentes entre ces objets. C'est cette faculté que la DRS s'efforce de remplacer.

La RIC implique deux étapes :

1. *La constitution de la BD contenant les images* : il s'agit de stocker les images elles-mêmes ainsi que les caractéristiques visuelles de chaque image dans des champs spécifiques de la BD. À l'heure actuelle, il est impossible d'émuler le jugement humain au sein d'un programme de description automatique d'image de manière satisfaisante, car il est très difficile de reproduire les interpolations que le cerveau

humain est capable de faire à partir d'une image plane (le concept d'*image plane* sera défini par la suite).

2. *La recherche d'image dans la BD* : qu'elle soit par définition de critères ou avec image de référence, il s'agit de comparer des caractéristiques d'entrée aux caractéristiques des images stockées dans la BD. De notre point de vue, ce qu'il est important de noter est que cela implique de stocker les informations sous un format permettant d'effectuer cette opération de comparaison facilement.

1.2.1 Positionnement du problème

Nous avons indiqué ci-dessus l'application sur laquelle nous avons basé notre travail, à savoir la RIC par similarité (RICS). De nombreux algorithmes de RICS ont déjà été publiés. La majorité utilise des critères de couleur, de forme, de texture ou de relations spatiales [3]. La recherche concernant l'extraction automatique de formes ou de contours pour ces traitements a déjà permis de produire des algorithmes exploitables [29], mais très peu pour l'extraction automatique de relations spatiales, bien que les applications soient déjà bien développées [10, 15]. La seule possibilité à l'heure actuelle est de définir les relations spatiales manuellement, et d'en tenir compte par la suite dans les algorithmes de recherche, ce qui est un travail généralement fastidieux, et certainement impossible à réaliser dans le cas de bases d'images de taille considérable comme celle de la NASA prise pour exemple un peu plus tôt. Des modèles théoriques ont été proposés [27, 6] afin d'obtenir les bases pour une implémentation, mais celle-ci n'a toujours pas été réalisée de manière satisfaisante à l'heure actuelle.

Pour clarifier un peu plus le domaine d'application du projet, il faut en outre préciser deux choses :

- L'étude porte exclusivement sur des **images**. Une suite à ce projet pourra être l'extension à d'autres types de supports multimédia comme la vidéo, la DRS pouvant par exemple permettre la description plus détaillée des scènes d'un film.
- En second lieu, le travail se concentre sur la proposition d'un algorithme capable de récupérer de façon systématique les relations spatiales entre les objets présents dans une image.

1.2.2 Les relations spatiales

Il est nécessaire d'apporter une précision sur ce que nous entendons par *relations spatiales* car il existe plusieurs types de relations entre deux objets dans une image. On peut distinguer deux catégories [10] :

Les relations directionnelles. Elles représentent les positions des objets les uns par rapport aux autres, ou bien par rapport au cadre de l'image, de la même manière que l'on exprime des directions géographiques par rapport aux quatre points cardinaux. En effet, il existe principalement trois types de relations directionnelles :
- strictes : Nord, Sud, Est et Ouest.
- mixtes : *idem* + Nord-Est, Nord-Ouest, Sud-Est et Sud-Ouest.
- de position : à droite, à gauche, au-dessus et en-dessous.

Ces relations sont décrites dans la figure 1.2.

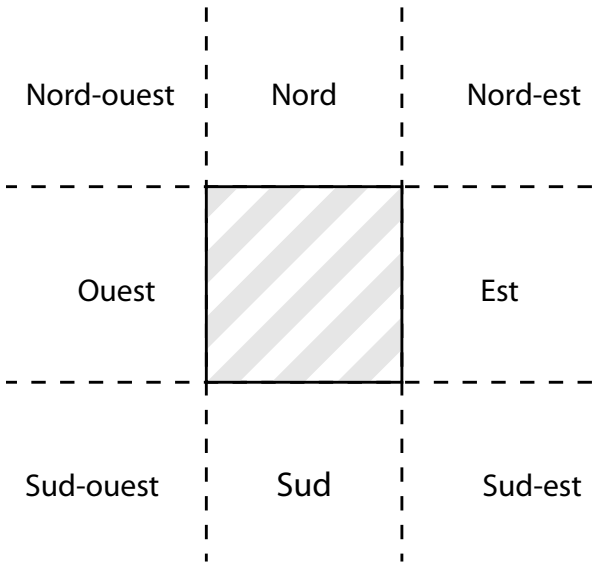


FIG. 1.2 – Relations directionnelles strictes et mixtes

Une autre possibilité utilisée dans la littérature est de déterminer ce type de positionnement à l'aide d'une *pente*, comme montré dans la figure 1.3. La position d'un objet A par rapport à un objet B est ainsi décrite par l'angle que fait la droite reliant les centres de ces deux objets par rapport à l'horizontale.

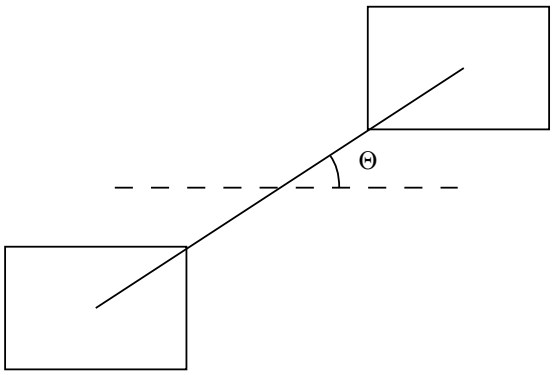


FIG. 1.3 – Relation directionnelle caractérisée par une pente

Les relations topologiques. Ces relations n'indiquent pas le positionnement cardinal des objets, mais plutôt l'interaction entre deux objets, c.-à-d. la manière dont un objet recouvre partiellement ou totalement un autre objet.

L'étude des relations spatiales représente un défi à l'heure actuelle et la solution à ce problème n'est pas triviale. Il serait irréaliste de prétendre pouvoir déterminer toutes les relations spatiales possibles entre des objets parfaitement déterminés tel que le ferait un être humain et il est préférable de s'intéresser plus profondément à un seul des aspects que nous venons d'évoquer plutôt que de s'étendre à la DRS généralisée. C'est pourquoi dans le cadre de ce mémoire nous nous concentrons sur les relations topologiques. Elles seront donc définies plus en détail dans le chapitre 2.

Chapitre 2

État de l'art

Dans ce chapitre, nous allons tout d'abord définir plus précisément ce que nous appelons des « objets », ensuite, nous allons apporter des précisions quant à la manière dont nous avons choisi de définir les relations spatiales (ou topologiques) entre les objets, et pour terminer nous verrons quels sont les prérequis nécessaires à la détection de relations spatiales.

2.1 Définition d'un « objet »

2.1.1 Notion d'objet en traitement d'image

Qu'entend-on en réalité par le terme « objet » dans le cas du traitement d'image ? C'est une question qui pose encore beaucoup de problèmes, et c'est un des axes de recherche les plus en vogue à l'heure actuelle. En effet, si un humain est capable de dire que l'image représente « un ballon rouge » (sur fond blanc par exemple), un programme informatique, lui, n'est capable que de dire que l'image représente « un *objet* de forme ronde et de couleur rouge, ayant une texture unie ». Ceci est un cas extrêmement simple, et on pourrait éventuellement se satisfaire de la description fournie par le programme. Toutefois, si l'on prend le cas du drapeau canadien sur un fond bleu uni tel que présenté sur la figure 2.1, les résultats donnés par un programme informatique sont loin d'être comparables avec l'interprétation humaine.

En effet, un programme informatique verra pour l'exemple du drapeau canadien : (*cf.* figure 2.2)

- un objet *n°1* rouge de forme rectangulaire,
- un objet *n°2* blanc de forme rectangulaire,
- un objet *n°3* rouge de forme rectangulaire,
- un objet *n°4* rouge de forme quelconque.

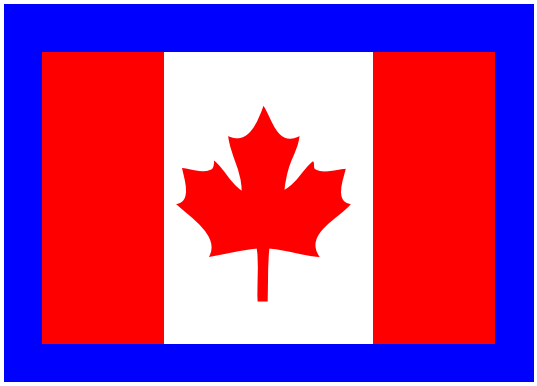


FIG. 2.1 – Drapeau canadien

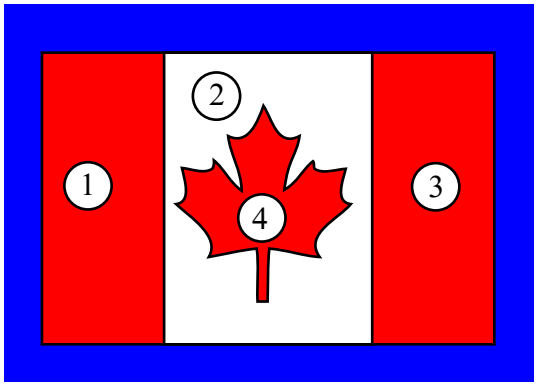


FIG. 2.2 – Drapeau canadien décomposé

Pour l'heure, le seul moyen d'identifier réellement les objets dans une image est d'effectuer une comparaison des objets segmentés à des objets bien connus stockés dans des bases de données prévues pour cela, comme présenté dans la section 1.1. Ce n'est pas le propos de ce mémoire, et nous préférons une définition plus basique pour mettre en place notre système.

2.1.2 Définition d'un objet

Dans le cadre de ce mémoire, nous définissons donc les « objets » comme étant des zones *homogènes* possédant un contour *fermé*. Nous reprenons donc la définition exploitable pour un programme informatique, et suivant cette définition, nous obtenons bien la liste énoncée ci-dessus pour l'exemple du drapeau canadien. De plus, l'*homogénéité* d'une région ne prendra en compte que des critères de couleurs dans notre cas. Une extension à des critères de texture pourra être envisagée par la suite.

Il est à noter qu'en général, les études théoriques concernant les RS distinguent trois types d'objets [9] : les points, les lignes et les régions. Pour le traitement d'image toutefois, cette distinction n'est pas applicable. On peut en effet considérer qu'un point est représenté par une région d'une taille limitée. Il en est de même pour une ligne, bien que la définition soit un peu plus complexe dans ce cas et devrait faire apparaître des critères d'épaisseur. Dans ce mémoire, nous nous concentrons sur la description d'objets sous forme de **régions**.

De plus, certains travaux ont étendu la définition des objets à des objets *complexes* [27], considérant par exemple des régions complexes composées de plusieurs régions simples et pouvant comporter des trous, comme montré sur la figure 2.3 [1].

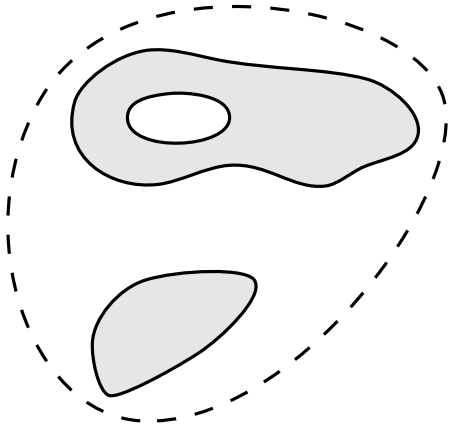


FIG. 2.3 – Exemple de région complexe

Cette définition pose des problèmes d'interpolation pour reconnaître les différentes parties d'un même objet, comme c'est le cas pour les quatres parties du cube rouge qui ne sont pas cachées par le rond bleu dans la figure 2.4. Cela fait intervenir des notions d'intelligence artificielle et d'apprentissage machine, et c'est donc situé en dehors du cadre de ce mémoire. Nous nous contenterons donc d'ignorer le concept d'objets complexes et de nous concentrer sur la récupération d'objets simples.

[1] Cette figure ne représente qu'**une seule** région complexe, qui est la réunion de deux régions simples et qui comporte un trou.

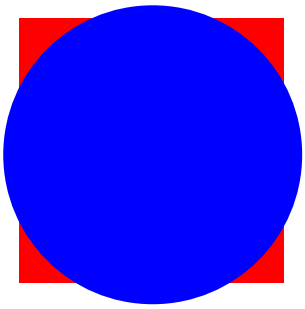


FIG. 2.4 – Exemple de cas d'interpolation pour une reconnaissance d'objet.

En résumé, nous considérerons comme un *objet* une zone **homogène en couleur**, **non multiple** et avec un **contour fermé**.

2.2 Définition des relations spatiales

Beaucoup d'études théoriques ont été menées sur la manière de définir des relations *spatiales* (ou *topologiques*) entre des objets dans un cadre bidimensionnel. La difficulté tient justement au fait que les images sont des environnements en deux dimensions, et si l'être humain est capable de reconstituer virtuellement un objet, ce n'est pas le cas d'un programme informatique. Ainsi, il est pour le moment quasi-impossible de faire dire à un programme informatique qu'un objet A est situé « devant » un objet B, ou que l'objet B est « caché » par l'objet A.

Il a donc fallu définir des relations identifiables par un programme informatique. Un premier modèle baptisé « modèle des quatre intersections » [7] a été proposé par M. J. Egenhofer. Il a ensuite été étendu à un modèle à neuf intersections [8, 5]. Ce modèle a servi de base à des études sur toutes les relations possibles entre différents types d'objets complexes [27]. L'étude citée de M. Schneider est la plus complète qui soit disponible sur les RS. Elle a permis de mettre en évidence les huit relations primaires qui sont détaillées dans la figure 2.5. Quatre de ces relations ont été regroupées par paires car ce sont des relations symétriques.

Il convient d'apporter une précision sur les deux couples de relations *couvre/est recouvert par* et *contient/est à l'intérieur de*. Ces deux paires de relations sont très semblables, comme le montre la figure 2.5. La différence réside uniquement dans le fait que pour la paire *couvre/est recouvert par*, les contours des deux objets ont une partie

commune, alors que pour la paire *contient/est à l'intérieur de*, les deux contours sont complètement disjoints.

Les autres relations sont suffisamment triviales pour se passer d'explication.

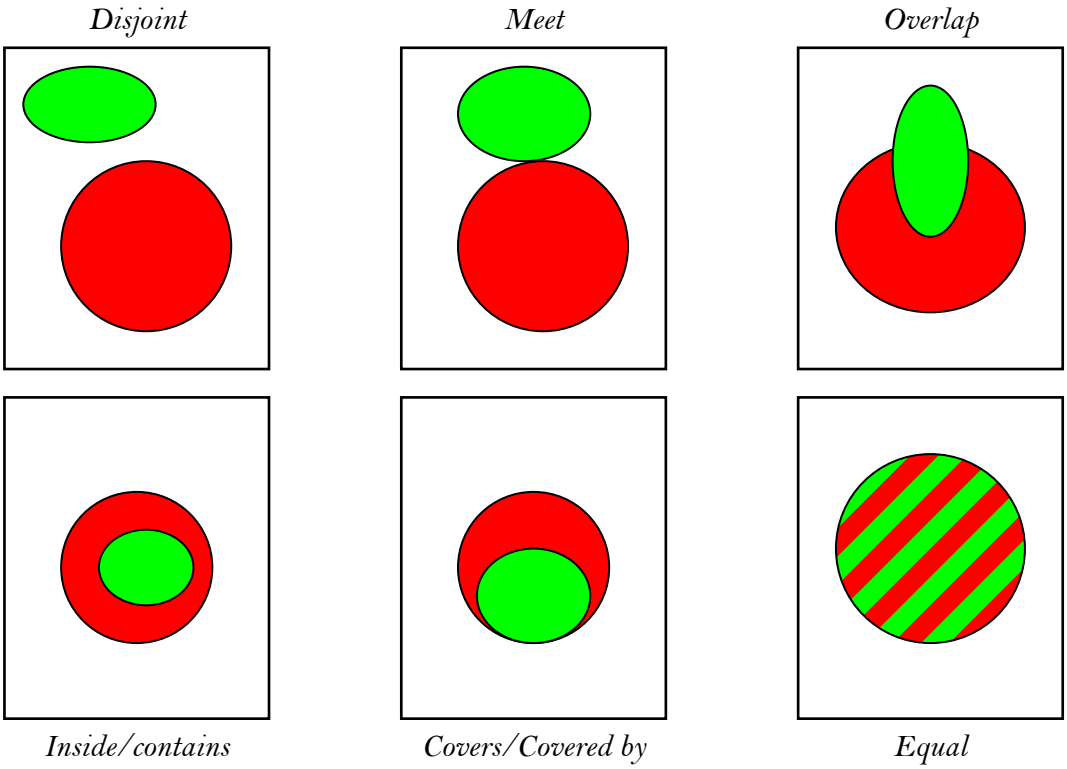


FIG. 2.5 – Les huit relations spatiales primaires

Les relations décrites dans la figure 2.5 portent leurs noms anglais d'origine. En voici une traduction :

Disjoint : disjoint	**Contains :** contient
Meet : touche	**Covers :** couvre
Overlap : chevauche	**Covered by :** est recouvert par
Inside : est à l'intérieur de	**Equal :** égal

TAB. 2.1 – Les huit relations primaires et leur traduction

Les définitions formelles de ces relations seront données dans la partie 3, ainsi que les adaptations nécessaires pour l'implémentation de ses définitions dans un modèle informatique.

2.3 Méthodes existantes de DRS

2.3.1 Méthode de Kim et Um

Kim et Um [17] Ont proposé en 1999 une des premières méthodes de DRS automatisée. Nous allons en voir les caractéristiques et les limites.

Caractéristiques de la méthode

Cette méthode s'appuie sur des objets de type *vectoriels*. La particularité de ces objets est qu'ils sont connus **entièrement**. Ainsi, les parties d'un objet A (par exemple) qui seraient cachées par un objet B, sont tout de même présentes dans l'image. Cela reprend la manière de travailler de certains logiciels d'imagerie qui utilisent des *calques vectoriels*. Chaque objet est contenu sur un calque différent, ce qui permet de récupérer séparément les objets de l'image. La notion de calque reste une *métaphore* puisqu'il s'agit en réalité d'une structure de données plus complexe, mais le principe est le même : chaque objet est défini séparément des autres, et dans sa globalité. Cette approche suppose une détection manuelle des objets au préalable.

Les auteurs utilisent l'approximation du plus petit rectangle conteneur (*minimum bounding rectangle*) de sorte que chaque objet est remplacé par le plus petit rectangle qui le contient. Les opérations géométriques sont ensuite effectuées directement sur les rectangles associés à chaque objet.

Ces opérations consistent à comparer les abscisses et les ordonnées des côtés des rectangles et d'inférer des relations spatiales. Dans l'exemple de la figure 2.6, on compare les bornes min et max en hauteur et en largeur des objets A et B. Ici, nous avons $\text{minY(B)} < \text{minY(A)} < \text{maxY(A)} < \text{maxY(B)}$. On en déduit donc que la bande horizontale contenant A est contenue dans la bande horizontale contenant B. De plus, nous avons aussi $\text{minX(A)} < \text{minX(B)} < \text{maxX(A)} < \text{maxX(B)}$, ce qui signifie que les deux bandes verticales contenant A et B se chevauchent. De ces deux relations, nous pouvons conclure qu'il s'agit d'un chevauchement.

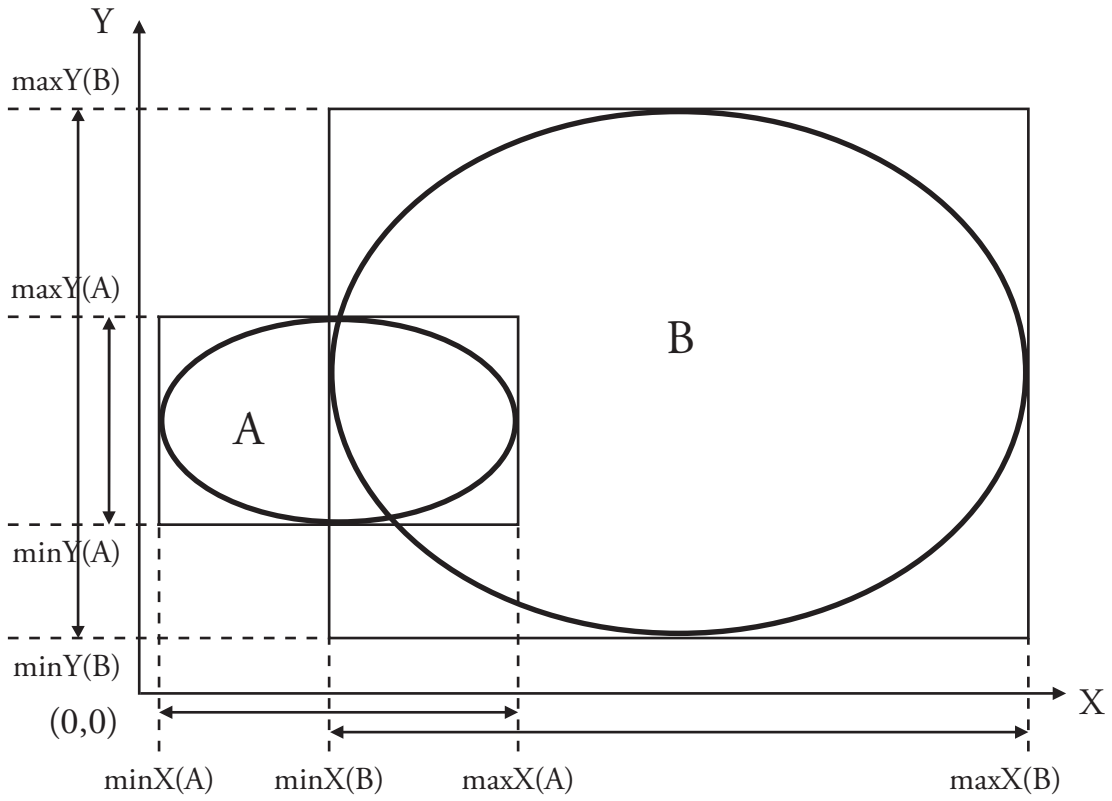


FIG. 2.6 – Illustration de la méthode de Kim et Um dans le cas d'un chevauchement

Critique de la méthode

Cette méthode possède des limites. Nous allons détailler ici les plus importantes :

1. l'approximation du plus petit rectangle conteneur est source de beaucoup d'erreurs de jugement. Par exemple, dans la figure 2.7, nous avons exactement les mêmes relations d'ordre que dans la figure 2.6, sans pour autant avoir la même relation : dans le cas de la figure 2.6, il s'agit d'un chevauchement alors que dans le cas de la figure 2.7 les deux objets sont disjoints.
2. Ce modèle pose aussi des problèmes pour la définition des relations *touche* et *couvre/est recouvert par*. On comprend facilement que le fait d'avoir une borne commune n'est pas une condition suffisante pour justifier d'avoir un contour commun, ce qui est une des caractéristiques définissant ces trois relations.

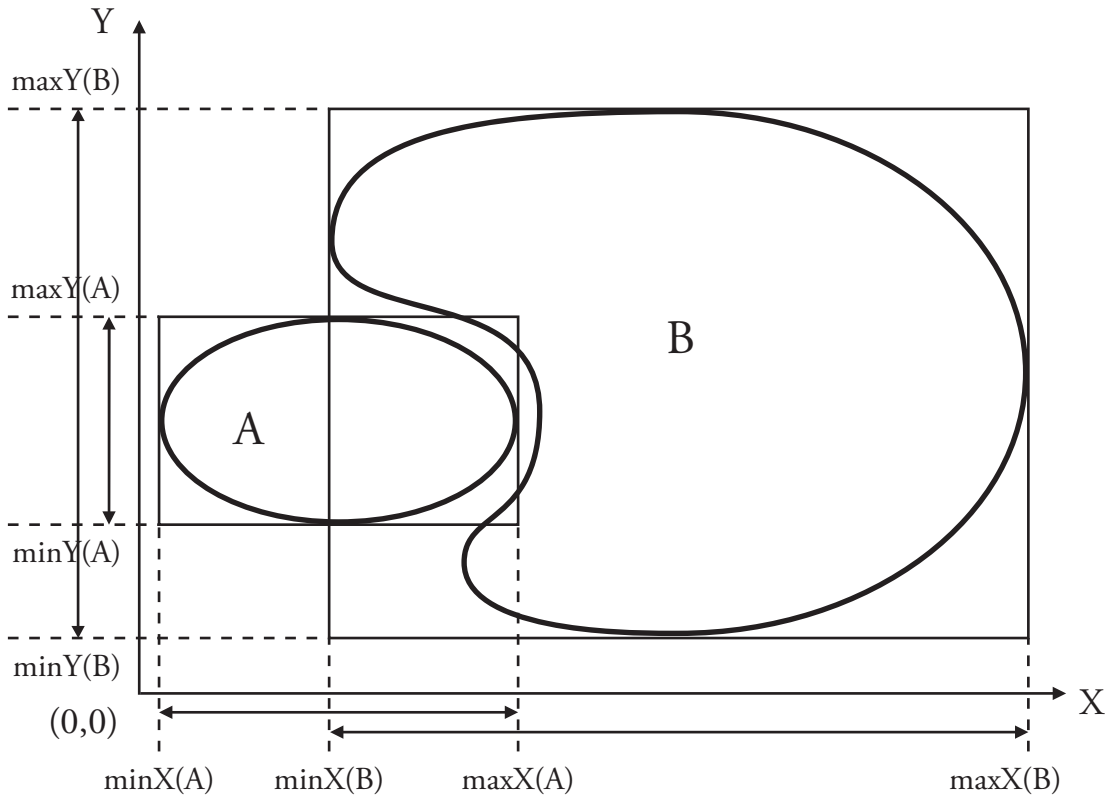


FIG. 2.7 – Confusion d'une disjonction avec un chevauchement

Ces erreurs de jugement font que cette méthode n'est pas fiable pour comparer des images entre elles. Elle peut toutefois s'avérer utile si on l'utilise en négation, c'est à dire pour déterminer si des images **ne sont pas** similaires.

2.3.2 Méthode de Walker et al.

Quelques méthodes ont été proposées pour effectuer des DRS, et surtout pour les représenter de manière efficace. On peut citer en particulier la représentation à l'aide de matrices de pondération mise au point par Walker et al. [30]. Cette méthode a été conçue principalement pour les systèmes de gestion de bases de données géographiques et s'appuie sur un calcul de distance entre les objets dans l'image étudiée. Les résultats des différents calculs de distances effectués au sein de l'image sont stockés dans une matrice carrée symétrique de taille égale au nombre d'objets présents dans l'image. L'élément $\mathcal{M}[i, j]$ est égal à la distance séparant l'objet i de l'objet j. La matrice ainsi

construite est appelée *matrice de contiguïté* et permet de déduire les relations spatiales.

Par exemple, pour le groupe d'objets présenté par la figure 2.8 :

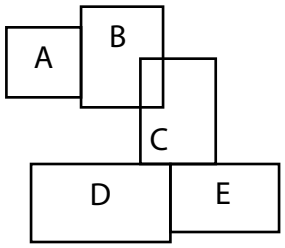


FIG. 2.8 – Exemple d'utilisation de la méthode de Walker et al.

On obtient la matrice suivante :

$$\mathcal{M} = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 0 & 0 \\ B & 1 & 0 & 2 & 0 & 0 \\ C & 0 & 2 & 0 & 1 & 1 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

Les valeurs composant la matrice sont les degrés de relation présents entre les différents couples d'objets. La force de chacun de ces degrés permet de déterminer la relation. Les auteurs définissent les correspondances suivantes :

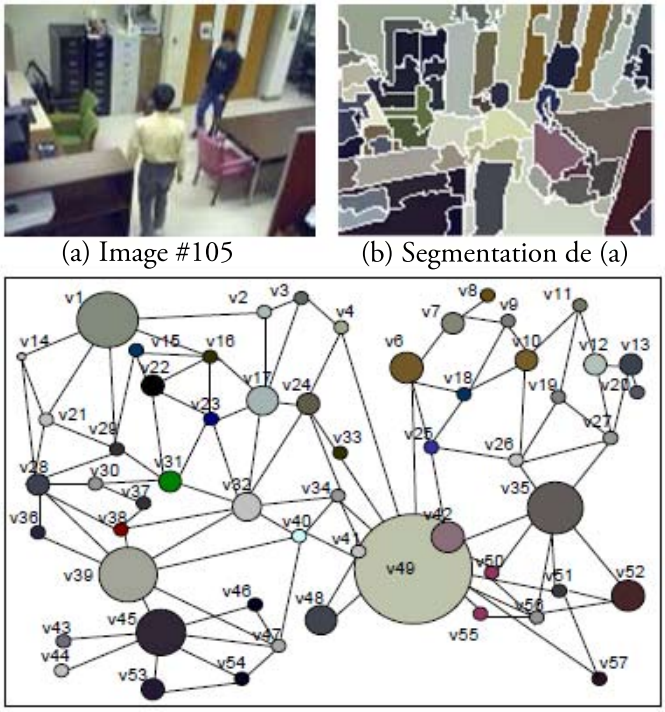
Poids	$\longrightarrow$	**Relation**
7		point commun
6		contient
5		chevauche
4		adjacent
3		chevauchement d'un des objets par le plus petit rectangle conteneur de l'autre
2		chevauchement des rectangles conteneurs
1		séparé
0		egal

Cette méthode diffère de celle que nous proposons à plusieurs points de vue : pour commencer, elle ne traite pas tout à fait des huit relations primaires, et nécessite la prise en compte de relations particulières mettant en jeu les plus petits rectangles conteneurs. Ensuite, le chevauchement pose le même problème que la méthode de Kim et Um : il nécessite de connaître non seulement les parties « visibles » des objets, mais aussi leurs parties « cachées », et ceci n'est pas applicable dans notre cas.

2.3.3 Méthode de Lee et al.

Plus qu'une méthode de DRS à proprement parler, Lee et al. [19] ont mis au point une méthode de représentation des relations spatiales. Ils fragmentent tout d'abord l'image en zones homogènes, de la même manière que nous le faisons dans notre méthode, puis établissent un graphe d'« adjacence ». Ce graphe permet de relier les régions adjacentes deux à deux, et de décrire ainsi l'image sous forme de relations d'objets.

Ils donnent l'exemple repris dans la figure 2.9 pour illustrer la méthode :



(a) Image #105 (b) Segmentation de (a)

(c) $Gr(I_{105})$ pour (b)

FIG. 2.9 – Application de la méthode de Lee et al.

La figure 2.9 montre une image, sa décomposition en objets et le graphe d'adjacence correspondant. Cette méthode permet de décrire une image efficacement, mais il est impossible à partir de cette méthode de déterminer des relations spatiales tel que nous voulons le faire : on ne peut par exemple pas déduire une relation de chevauchement à partir du graphe obtenu. Cette méthode est donc similaire de par sa finalité qui est aussi la description des relations entre les objets d'une image, mais elle diffère par la forme de ce que nous voulons mettre en place.

2.4 Travaux connexes

Cette dernière section va nous permettre de voir quelques autres travaux portant sur la DRS, et les raisons par lesquelles ils ont motivé notre approche.

Tout d'abord, Nabil et al. [23] ont eux aussi porté leur recherche sur la manière d'améliorer la RIC en prenant en compte les relations spatiales dans les requêtes. Leur système, toutefois, se concentre sur un aspect différent de celui qui nous a intéressé, à savoir les conditions de similarité entre les images au niveau des relations spatiales. Ils ne se sont en effet pas intéressés à la détection de relations spatiales dans l'image, qu'ils se contentent de décrire manuellement, mais plutôt à la manière d'utiliser ces relations spatiales par la suite dans les requêtes. Ils sont partis du principe que l'on pouvait classer des images mettant en jeu les mêmes objets selon la proximité des relations spatiales. Par exemple, dans la figure 2.10, la première image est plus proche de la seconde que de la troisième, car la relation « à l'intérieur de » est plus proche de la relation « chevauche » que de la relation « est disjoint de ».

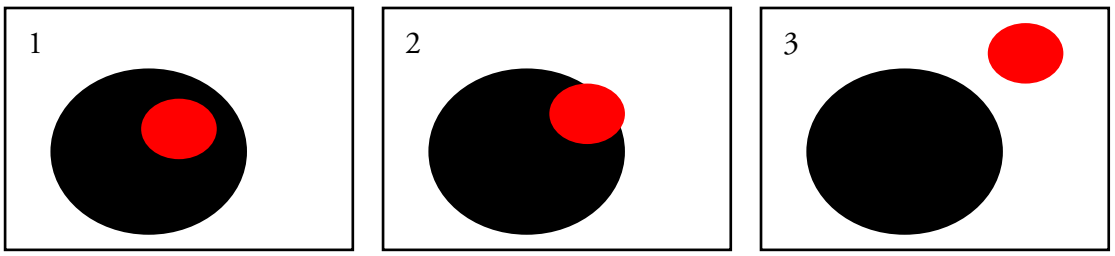


Fig. 2.10 – Ressemblance des relations spatiales

Ils ont ainsi établi un graphe de proximité des relations spatiales qui peut être utilisé dans une méthode de RIC tenant compte des relations spatiales. Ces travaux sont donc

complémentaires aux nôtres : ils pourraient permettre par la suite d'implémenter un Système de RIC permettant à la fois de détecter les relations spatiales automatiquement dans les images (ce que nous faisons) et ensuite de faire des recherches raffinées selon ces relations spatiales (ce sur quoi on travaillé Nabil et al.).

Ensuite, Chang et al. [23] ont mis au point un système de comparaison d'images *icôniques* [2]. Il fonctionne sur le principe d'un calcul de distance entre les représentations icôniques des images. Il est par contre plutôt prévu pour traiter les relations directionnelles que les relations topologiques et quoiqu'il en soit, il se fonde là encore sur une détermination manuelle des relations spatiales.

À l'heure actuelle, les recherches portent donc plutôt sur les moyens d'utiliser les relations spatiales dans un système de recherche que que sur ceux de les détecter dans les images de manière automatique. C'est pour cette raison que nous avons choisi de nous intéresser plutôt à l'aspect de détection automatique des relations spatiales, qui permettrait de compléter le processus complet de traitement des RS dans les base de données d'images.

[2] Traduit de l'anglais "iconic images", le terme représente en réalité des structures de données contenant une description de l'image sous forme d'un ensemble d'objets et de relations entre ces objets.

Chapitre 3

Méthodologie

Dans ce chapitre, nous allons d'abord énoncer les objectifs de recherche à atteindre. Ensuite, nous verrons le raisonnement qui nous a permis d'aboutir à la conception d'une méthode permettant de réaliser ces objectifs. Pour finir, nous décrirons la méthode elle-même.

3.1 Objectifs du projet

Cette partie a pour but d'énoncer les objectifs fixés dans cette recherche et d'indiquer la démarche suivie pour atteindre de tels objectifs.

Nous avons présenté dans la section 2.2 les huit relations spatiales considérées comme « primaires » dans la littérature. Le but recherché lors de la définition de ces huit relations primaires était d'obtenir un ensemble restreint de RS pouvant ensuite être dérivées pour produire n'importe quelle autre RS. Ces relations constituent donc le point de départ de tout travail sur les relations spatiales, et logiquement, elles servent de fil conducteur à notre projet : le but final de ce mémoire est donc de développer une procédure – ou un ensemble de procédures – permettant d'identifier chacune de ces huit RS primaires de manière autonome.

L'autonomie dont nous parlons ici caractérise le fait qu'aucune intervention extérieure n'est requise. Tout le processus doit se faire dans la transparence pour l'uti-

lisateur, contrairement aux systèmes actuels pour lesquels la définition des relations spatiales se fait encore manuellement.

Par ailleurs, le prétraitement qui consiste à détecter les objets dans l'image et à les isoler pour pouvoir déterminer les relations existant entre eux doit respecter certaines contraintes que nous allons voir ci-dessous. Nous l'avons donc intégré à notre méthode, mais comme nous l'avons précisé dans la partie 2.1, il pourra être remplacé dans le futur par un procédé spécifique plus performant. La méthode générale de DRS que nous proposons est en effet indépendante du processus de segmentation de l'image en objets.

Un utilisateur n'a ainsi qu'à se soucier de passer en entrée une image respectant un certain format. Du point de vue de notre système et du fait de la simplicité du processus de segmentation que nous utilisons, le format bitmap est conseillé pour éviter la pollution due à la compression des autres formats, en particulier JPEG, bien que ces autres formats soient acceptés. La méthode produit ensuite en sortie une structure de données représentant les relations spatiales extraites de l'image. Cette structure de donnée sera choisie parmi celles proposées dans la littérature, telles que la représentation par *2D-String* de Chang et al. [2] ou encore la représentation par *θR-String* de Gudivada [14].

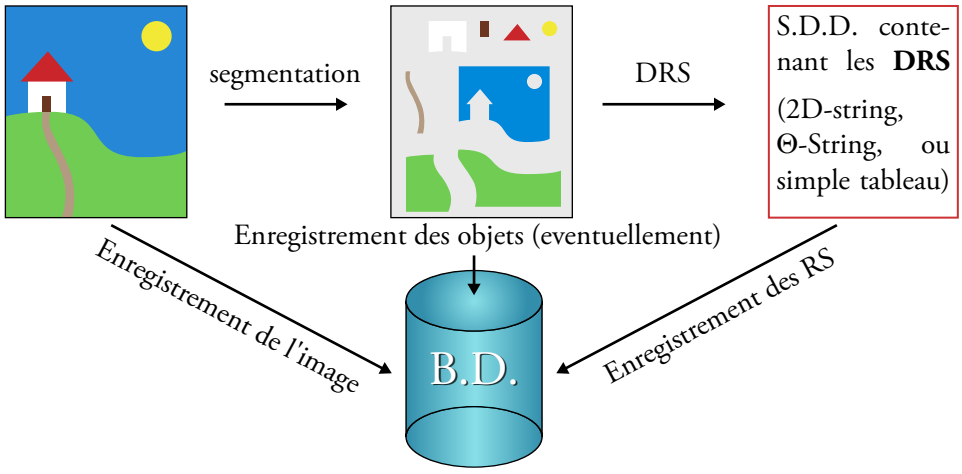


FIG. 3.1 – Structure générale du procédé

La figure 3.1 montre la structure générale du traitement d'une image en vue de l'intégrer à une BD telle que nous la concevons dans ce mémoire. La segmentation fait partie du projet mais c'est la DRS (en rouge) qui en est le cœur.

3.2 Problèmes posés

3.2.1 Le modèle théorique

Chacune des huit relations primaires que nous avons précédemment nommées a été formellement définie par M. Schneider et T. Behr [27] de la manière suivante :

Soit un objet A. On pose les notations suivantes :

Notation 1 : ∂A représente le contour de A ;

Notation 2 : A° représente l'intérieur de A ;

Notation 3 : A^- représente l'extérieur de A.

En étendant ces notations à un objet B, on peut définir ainsi les huit relations primaires possibles entre A et B :

disjoint(A,B)	$A^\circ \cap B^\circ = \varnothing \wedge A^\circ \cap \partial B = \varnothing \wedge \partial A \cap B^\circ = \varnothing \wedge \partial A \cap \partial B = \varnothing$
meet(A,B)	$A^\circ \cap B^\circ = \varnothing \wedge (A^\circ \cap \partial B \neq \varnothing \vee \partial A \cap B^\circ \neq \varnothing \vee \partial A \cap \partial B \neq \varnothing)$
inside(A,B)	$A^\circ \cap B^\circ \neq \varnothing \wedge A^\circ \cap B^- = \varnothing \wedge A^- \cap B^\circ \neq \varnothing \wedge \partial A \cap \partial B = \varnothing$
contains(A,B)	$A^\circ \cap B^\circ \neq \varnothing \wedge A^\circ \cap B^- \neq \varnothing \wedge A^- \cap B^\circ = \varnothing \wedge \partial A \cap \partial B = \varnothing$
coveredBy(A,B)	$A^\circ \cap B^\circ \neq \varnothing \wedge A^\circ \cap B^- = \varnothing \wedge A^- \cap B^\circ \neq \varnothing \wedge \partial A \cap \partial B \neq \varnothing$
covers(A,B)	$A^\circ \cap B^\circ \neq \varnothing \wedge A^\circ \cap B^- \neq \varnothing \wedge A^- \cap B^\circ = \varnothing \wedge \partial A \cap \partial B \neq \varnothing$
equal(A,B)	$A^\circ \cap \partial B = \varnothing \wedge A^\circ \cap B^- = \varnothing \wedge \partial A \cap B^\circ = \varnothing \wedge$ $\partial A \cap B^- = \varnothing \wedge A^- \cap B^\circ = \varnothing \wedge A^- \cap \partial B = \varnothing$
overlap(A,B)	$A^\circ \cap B^\circ \neq \varnothing \wedge A^\circ \cap B^- \neq \varnothing \wedge A^- \cap B^\circ \neq \varnothing$

TAB. 3.1 – Définitions formelles des huit relations primaires [27]

Pour transposer ces définitions sous forme d'algorithmes, on peut commencer par les synthétiser sous la forme d'un tableau représentant la valeur des intersections de

chacune des composantes de l'objet A (c.-à-d. l'extérieur, l'intérieur et le contour) avec les composantes de l'objet B. C'est ce qui est fait dans le tableau 3.2

disjoint(A,B)

	∂A	A°	A^-
∂B	0	0	-
B°	0	0	-
B^-	-	-	-

meet(A,B)

	∂A	A°	A^-
∂B	1	-	-
B°	-	0	-
B^-	-	-	-

inside(A,B)

	∂A	A°	A^-
∂B	0	-	-
B°	-	1	1
B^-	-	0	-

contains(A,B)

	∂A	A°	A^-
∂B	0	-	-
B°	-	1	0
B^-	-	1	-

coveredBy(A,B)

	∂A	A°	A^-
∂B	1	-	-
B°	-	1	1
B^-	-	0	-

covers(A,B)

	∂A	A°	A^-
∂B	1	-	-
B°	-	1	0
B^-	-	1	-

equal(A,B)

	∂A	A°	A^-
∂B	-	0	0
B°	0	-	0
B^-	0	0	-

overlap(A,B)

	∂A	A°	A^-
∂B	-	-	-
B°	-	1	1
B^-	-	1	-

TAB. 3.2 – Intersections des composantes de A et B pour chaque relation [27]

Un "0" dans une cellule signifie que l'intersection doit être vide, un "1" signifie qu'elle doit être non vide, et un tiret indique qu'elle peut être quelconque.

3.2.2 Les limites du modèle

Le tableau 3.2 est en réalité une représentation matricielle du modèle des neuf intersections [4]. Ce modèle pose un problème majeur : il faut pouvoir identifier les trois composantes : intérieur, extérieur et contour pour chaque objet, et étant donné que nous travaillons sur des images planes [1], les parties cachées des objets ne sont pas connues, que ce soit les parties de l'intérieur ou bien les parties du contour. On ne peut donc pas appliquer directement le modèle des neufs intersections décrit ci-dessus pour obtenir les relations spatiales. Il faut émuler certaines caractéristiques de ce modèle à l'aide de caractéristiques récupérables sur des objets non entièrement connus. Cela entraîne nécessairement une divergence par rapport au modèle original, mais ce n'est pas un problème en soit, le principal étant de s'approcher au mieux de la perception humaine.

Pour chaque objet, nous disposons d'un nombre limité d'informations exploitables pour la recherche de relations spatiales :

1. Tout d'abord, les informations sur le contour des objets : il s'agit de récupérer des indications locales sur les contours extérieurs et intérieurs (le cas échéant) d'un objet. Ces informations comprennent la taille du contour (en nombre de pixels), le nombre de contours voisins et l'identifiant de ces contours, et plus particulièrement, la taille de la zone de contact.
2. Ensuite, un jeu d'inférences (ex : transitivité de la relation de contenance) nous permet de compléter le modèle (cf. détails dans la partie 3.6).

En théorie, on devrait simplement comparer les résultats obtenus avec les matrices décrites dans le tableau 3.2, pour finalement déduire les relations spatiales. Malheureusement, comme indiqué précédemment, ce modèle n'est pas applicable en tant que tel. En particulier, les informations que nous pouvons extraire d'une image plane ne nous permettent pas à proprement parler de définir des *intersections*. Nous devons donc déduire les relations spatiales directement des informations extraites.

[1] Nous appelons dans ce mémoire « image plane » une image matricielle, par opposition aux images vectorielles ne contenant qu'une seule instance de chacune des couches qui la composent (typiquement : trois couches pour une image en couleur, et une couche pour une image en niveaux de gris. De manière générale, il s'agit des images ne contenant pas de calques comme il est maintenant possible de le faire à l'aide de certains logiciels d'imagerie.

Ces observations nous ont permis de mettre au point le processus de DRS qui sera présenté dans la section suivante.

3.3 Technique générale

Le processus de détection des relations spatiales tel que nous l'envisageons peut être décomposé en trois parties :

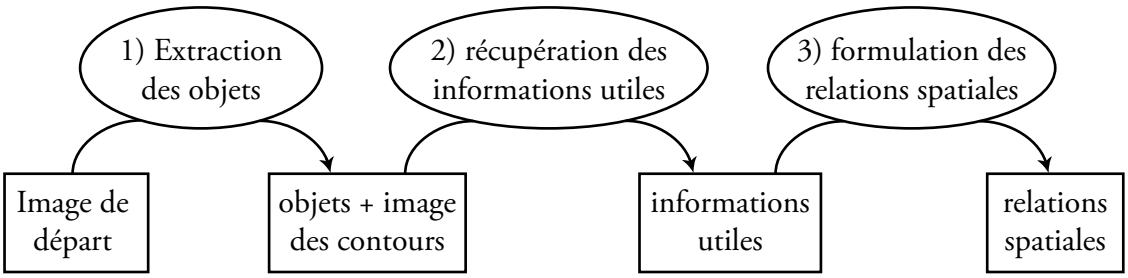


FIG. 3.2 – Technique générale

1. L'isolement de chaque objet présent dans l'image et son identification. Dans notre cas, cela passe tout d'abord par la décomposition de l'image en zones de couleur homogène d'après la définition que nous avons donnée dans la partie 2.1. Il peut y avoir plusieurs zones d'une même couleur, il faut donc bien faire la différence entre une décomposition de l'image selon les couleurs, et une décomposition de l'image en objets. Le nombre de couleurs présentes dans l'image sera toujours inférieur ou égal au nombre d'objets qui la composent.

   Ensuite, il faut récupérer les contours des objets tout en gardant l'identification. En effet, un objet aura toujours au minimum un contour externe, mais pourra aussi avoir un ou plusieurs contours internes. Il faut donc être capable d'établir le lien entre tous les contours d'un même objet.
2. La seconde partie consiste, pour chaque objet, à récupérer les informations pertinentes suivantes :
   - les objets auxquels il est relié. Pour cela, on procède à un *suivi des contours* (la méthode sera expliquée par la suite) pour chacun des contours de l'objet. On doit recueillir pour chaque contour suivi les identifiants des différents objets qui le bordent

– la nature des liens (ex : longueur de la zone de contact, contact selon un contour extérieur ou intérieur). Ces informations sont là encore collectées lors du suivi des contours.

3. Pour terminer, on utilise les informations ainsi obtenues pour déterminer les relations existant entre chaque objet présent dans l'image. Dans le cas de la figure 3.3 par exemple, en suivant le contour de l'objet bleu (n° 1), on découvre qu'il n'est bordé que par un contour de couleur rouge (celui de l'objet n° 2), et que la longueur du contour rouge est légèrement inférieure à celle du contour bleu. De ceci on peut déduire que l'objet n° 2 est *à l'intérieur de* l'objet n° 1, et que l'objet n° 1 *contient* l'objet n° 2.

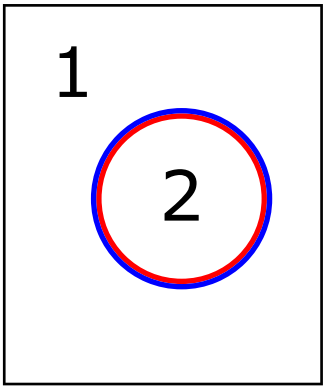


FIG. 3.3 – Exemple de détermination de la relation "contient / est à l'intérieur de"

3.4 Identification des objets

L'identification des objets vise à préparer l'image pour la détection des relations spatiales. Comme nous l'avons dit, c'est un processus indépendant de la DRS en elle-même. Toutefois, le résultat en sortie doit respecter certaines contraintes.

Tout d'abord, nous ne traitons pas directement l'image en tant que tel, mais sa représentation matricielle. Nous travaillons donc sur un tableau. Ensuite, le tableau issu de ce pré-traitement et qui sera utilisé pour la DRS doit avoir les caractéristiques suivantes pour convenir à notre méthode :

– L'image qu'il représente ne contient plus les objets entiers, mais uniquement leurs contours. toutes les cases du tableau représentant des pixels situés sur la partie *intérieure* des objets doivent donc être vides.
– Ces contours doivent permettre l'identification des objets de manière unique. Ils doivent donc être marqués.

Enfin, ce traitement doit retourner en plus du tableau une structure de données contenant les caractéristiques de chaque objet, ces mêmes caractéristiques qui ont été utilisées pour effectuer la segmentation de l'image. Dans notre cas, il s'agit de la couleur, mais les caractéristiques peuvent être plus complexes et multiples. On pourrait prendre en compte en plus de la couleur la texture des objets, leur forme et éventuellement leur nom si l'on a procédé à une reconnaissance.

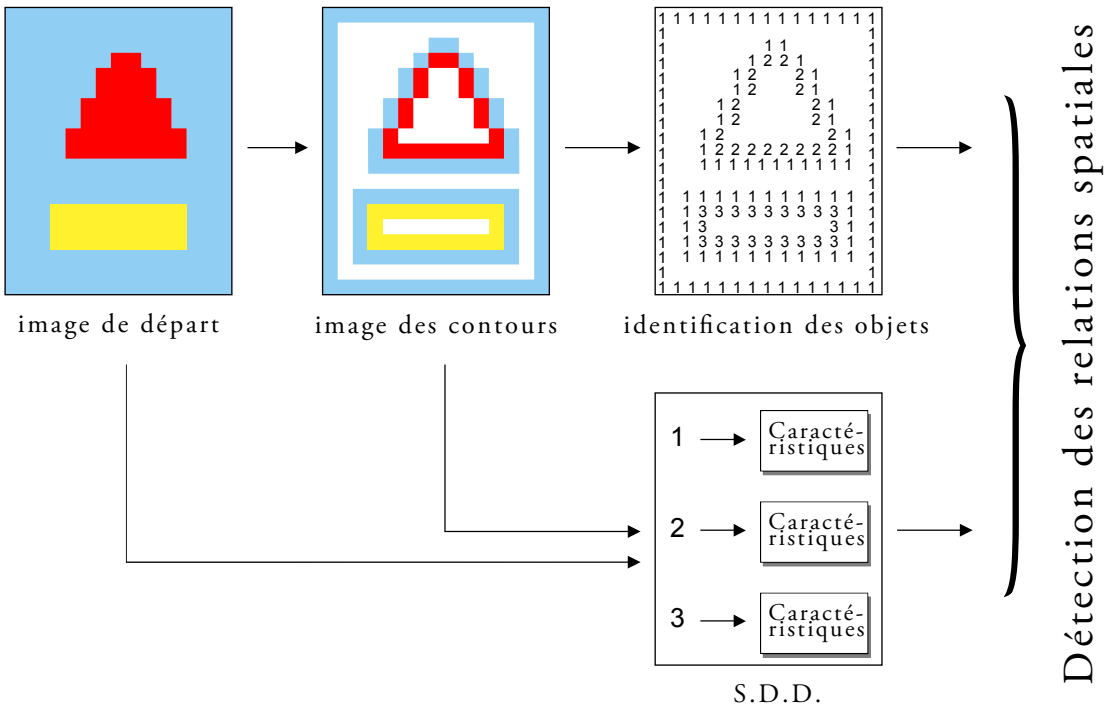


FIG. 3.4 – Structure du pré-traitement

La figure 3.4 représente le processus de pré-traitement tel que nous le concevons, et indique les données que récupère en entrée le processus de détection des relations spatiales.

Les trois parties suivantes vont maintenant nous permettre de détailler chacune des étapes de ce pré-traitement.

3.4.1 Segmentation de l'image en zones de couleur

Tout d'abord, l'image est segmentée en zones de couleurs homogènes.

Il faut noter deux points au sujet de la segmentation de l'image selon les couleurs :

– À chaque couleur détectée est associée une image différente. Cette image ne contient que les pixels de la couleur correspondante, le reste de l'image est gardé vide. À titre d'exemple, la figure 3.5 montre la segmentation d'une image selon les plages de couleurs. Le fond gris représente la partie « vide » de chaque « sous-image », et l'image qui contient normalement le gris est entièrement vide puisqu'il n'y a pas de gris dans l'image.

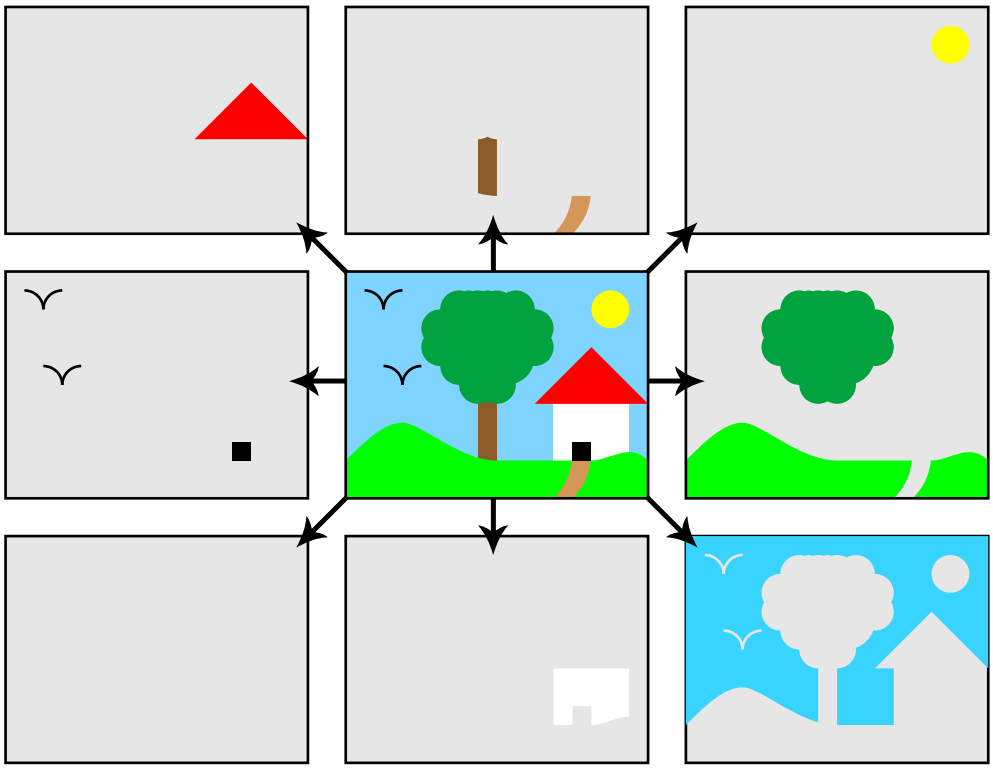


FIG. 3.5 – Exemple de segmentation selon des plages de couleur

– Ces nouvelles images créées peuvent contenir plusieurs objets au sens où nous l'entendons. En effet, il se peut que plusieurs objets de l'image soient de la même couleur. Il faut donc bien noter qu'à ce moment là, on ne parle pas encore réellement d'**objets**, mais bien de séparation des **couleurs**.

3.4.2 Récupération des contours

Une fois les zones de couleur séparées, on les « vide » pour ne garder que leurs contours. Cette opération nous permet de gagner en efficacité lors de la récupération des informations au niveau des contours, car les pixels ainsi vidés n'auront qu'à être tout simplement ignorés par la suite. Les informations de couleur seront utilisées pour différentier deux objets voisins d'un même objet lors du suivi des contours. C'est pour cette raison qu'on laisse les contours en couleur plutôt que de les représenter en noir et blanc, comme c'est l'usage.

Les images contenant chacune des couleurs sont ensuite concaténées en une seule image. Ici, le but est de n'avoir qu'une seule matrice à traiter par la suite, plutôt que de s'encombrer avec autant de matrices qu'il y aura d'objets segmentés. Cette opération n'est pas du tout gênante car les zones segmentées sont disjointes, l'image de retour a donc les caractéristiques suivantes :

– la plus grande partie de l'image est vide (transparente), seuls les contours des différents objets sont visibles.
– pour chaque objet détecté, l'image comporte un contour de la couleur de l'objet.
– à l'exception du fond de l'image qui a un contour d'un pixel d'épaisseur et de la même couleur que le fond le long des bords de l'image, les contours sont en réalité *doubles*. En effet, chaque objet étant toujours en contact avec un autre objet (le fond est considéré comme un objet), le contour de chaque objet est associé à celui ou ceux du (des) objet(s) adjacent(s).

C'est **cette image** qui sert de base pour la DRS en elle-même. Il faut donc une image avec des contours les plus nets possibles.

Un exemple de résultat du traitement décrit ci-dessus est donné dans la figure 3.6.

Les rayures au fond de l'image de droite signifient que les pixels du fond sont vides. Par ailleurs, on remarque bien qu'à l'exception de la bordure de l'image où il n'y a qu'un seul contour blanc (de la couleur du fond), tous les contours sont doubles.

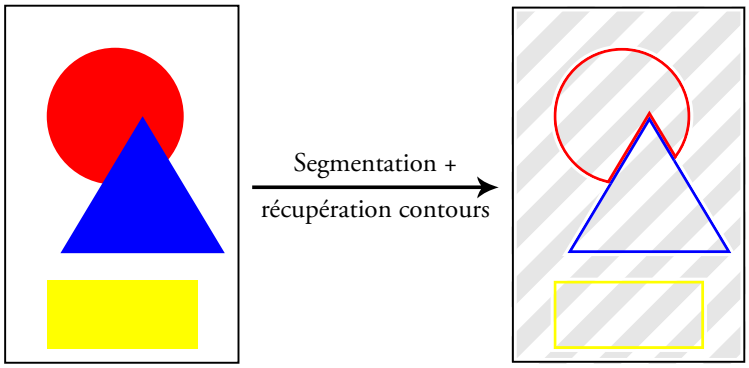


FIG. 3.6 – Exemple de segmentation + récupération de contours des objets

3.4.3 Attribution des identifiants

En prenant la représentation matricielle de l'image des contours, nous procédons à une identification des objets. Il est à noter qu'à partir d'ici, nous ne travaillons en réalité plus sur des représentations matricielles d'images, mais bien sur des tableaux d'entiers. De tels tableaux contiennent certaines valeurs fixées arbitrairement (tel que "-1" pour un pixel vide) qui ne correspondent pas obligatoirement à des informations visuelles et ces tableaux ne sont donc pas forcément affichables tels quels sous forme d'images.

Deux procédures sont concernées ici. La première est le suivi des contours. Il s'agit de créer un nouveau tableau contenant non plus les informations de couleur de chacun des contours, mais un identifiant qui leur est attribué. La seconde a pour but de donner le même identifiant à tous les contours d'un même objet, car il s'agit d'identifier les objets et non simplement les contours. Ces deux procédures seront détaillées dans le chapitre 4.

On obtient donc en sortie un tableau d'entiers contenant "-1" pour toutes les cases représentant des pixels vides, et l'identifiant de l'objet pour les cases correspondant à un pixel appartenant au contour d'un objet. ce résultat est représenté par la figure 3.7. Les rayures grises indiquent là encore des pixels « vides » et l'absence de chiffre dans le tableau de droite caractérise des cases vides (elles contiennent en réalité un zéro, mais il n'est ps indiqué sur la figure pour plus de lisibilité).

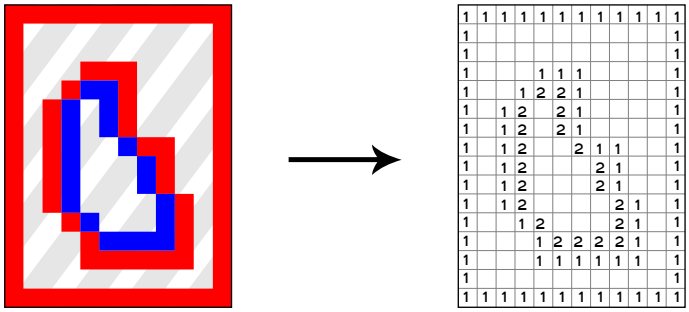


FIG. 3.7 – Résultat de l'identification des objets

Le processus de DRS va donc recevoir en entrée deux tableaux, l'un représentant les contours en couleur des objets, et l'autre contenant les identifiants des objets. Comme nous l'avons déjà signalé dans la partie 3.1, on pourrait, à la place du tableau contenant les contours en couleurs, passer en entrée de la DRS une structure de données plus complexe contenant toutes les informations pertinentes sur chaque objet. En ce qui nous concerne, la seule information que nous retenons des objets est leur couleur. Ainsi, le tableau représentant les contours en couleur est tout à fait suffisant.

3.5 Récupération des informations utiles : suivi des contours

Nous partons ici du fait que nous voulons récupérer pour chaque contour :

- sa longueur ;
- les identifiants de chacun des objets qui le bordent ;
- les longueurs des zones de contact.

Nous voulons de plus savoir s'il s'agit du contour extérieur de l'objet étudié ou bien d'un contour intérieur. Ceci part d'une constatation très simple : tout les objets $O_1...O_n$ qui ont un contour commun avec le contour intérieur d'un objet A sont nécessairement *contenus* dans A, et d'autre part, A *contient* tous les objets $O_1...O_n$ qui ont un contour commun avec un de ses contours intérieurs le cas échéant.

En sortie de ce traitement, nous devons produire une liste d'objets et une liste de contours. La structure de données représentant un objet contient les informations suivantes :

- son identifiant ;
- ses caractéristiques issues de la segmentation. Dans notre cas, il s'agit de sa couleur ;
- une liste de tous les contours qu'il possède.

Les contours sont eux aussi représentés par une structure de données particulière. Celle-ci contient :

- l'identifiant de l'objet auquel le contour appartient ;
- les caractéristiques de l'objet (les mêmes que dans la SDD représentant l'objet telle que nous venons de la voir) ;
- la longueur du contour ;
- un tableau contenant pour chacun des objets de l'image la longueur qu'il a en commun avec lui ;
- un booléen indiquant s'il s'agit d'un contour intérieur ou non. Le booléen est faux s'il s'agit d'un contour extérieur et vrai si c'est un contour intérieur.

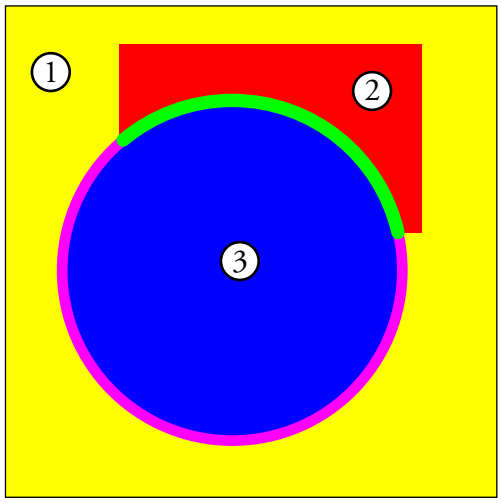


FIG. 3.8 – Exemple d'informations recueillies

Pour le contour de l'objet rond et bleu de la figure 3.8 par exemple, les informations seraient les suivantes :

Information recueillie	**Valeur de l'information**
• identifiant de l'objet auquel le contour appartient	3
• caractéristiques de l'objet	couleur : *rouge*
• longueur du contour	somme des longueurs des traits vert et magenta
• tableau contenant pour chacun des objets de l'image la longueur du contour commun, le cas échant	Objet 1 : longueur du trait magenta Objet 2 : longueur du trait vert
• booléen indiquant s'il s'agit d'un contour intérieur ou non	*false* (l'objet ne possède qu'un contour externe)

TAB. 3.3 – Exemple d'informations recueillies

Ces deux listes nous permettent ensuite de déterminer les relations spatiales tel que décrit ci-après.

3.6 Formulation des relations spatiales

La détermination des relations spatiales se fait par élimination, en commençant par les plus simples. Ceci permet de gagner en efficacité car les conditions les plus simples à déterminer sont vérifiées plus souvent que les conditions plus compliquées.

Par ailleurs, certaines relations sont symétriques, cela nous permet de déterminer certaines relations par paires. Par exemple, si l'on vient de déterminer qu'un objet **A** *contient* un objet **B**, alors on peut directement affirmer que **B** *est contenu dans* **A**. L'autre paire concernée est la paire *recouvre*/*est recouvert par*. Le processus de détermination des relations spatiales ne détermine par ailleurs pas *toutes* les relations, il doit être complété par la suite par certaines inférences. Par exemple, la relation *disjoint* implique qu'il n'y a pas de contact entre les deux objets, et il faut pour cela prendre en considération l'objet dans sa totalité. On ne peut donc pas se contenter d'étudier un de ses contours localement. Nous avons préféré déterminer cette relation en dernier lieu

après l'élimination de toutes les autres possibilités : si aucune des six premières relations ne se vérifie entre deux objets, alors on dira qu'ils sont nécessairement disjoints.

Pour terminer dans les hypothèses, nous devons préciser ici qu'en réalité, nous ne traitons que sept relations, et non huit comme le suggère le modèle que nous avons présenté dans la section 2.2. En effet, la relation d'égalité est aberrante dans le cas d'une image plane. Si deux objets sont égaux, alors nous n'en voyons qu'un seul, et il n'y a pas lieu de déterminer une relation.

les relations sont finalement déterminées dans l'ordre suivant :

1. contient / est contenu ;
2. touche ;
3. se chevauche avec ;
4. recouvre / est recouvert par ;
5. est disjoint de.

Cet ordre provient de celui dans lequel sont vérifiés les différents critères pris en compte. Ces critères sont classés par ordre de simplicité de vérification : les relations *contient / est contenu* ne nécessitent que la vérification d'un booléen. Les relations *se chevauche avec* et *touche* posent un problème. En effet, la seule chose qui différencie ces deux relations, du point de vue de notre méthode, est la longueur de la zone de contact. Il faut donc fixer un seuil de différenciation. Ceci est fait dans la partie 4, car il faut procéder à des expérimentations afin de simuler au mieux le jugement humain. Ces deux relations nécessitent donc la comparaison des valeurs des différentes longueurs concernées (cf. figure 3.8) avec les seuils fixés. Il en est de même pour les relations *recouvre / est recouvert par* qui font elles aussi intervenir des seuils, mais sur toutes les longueurs concernées cette fois. La relation *est disjoint de* est finalement attribuée à tous les couples d'objets restants après la vérification des autres relations. Tous ces critères apparaissent en détail dans l'algorithme 1.

En plus de la procédure décrite par cet algorithme, il faut rajouter un post-traitement visant à compléter la détermination des relations spatiales. C'est lors de ce post-traitement qu'est par exemple déterminée la relation *disjoint*, car comme nous l'avons dit, nous marquons comme étant une relation de disjonction toute relation non encore

Algorithme 1 : Détermination des relations spatiales

Entrées : Liste des objets, liste des contours
Sorties : Liste des relations spatiales
pour *i allant de 0 à "taille de la liste des contours - 1"* **faire**
  si *contour* i *est un contour intérieur* **alors**
    l'objet auquel appartient le contour i **contient** chacun des objets ayant un contour adjacent au contour i;
    chaque objet ayant un contour adjacent au contour i **est contenu par** l'objet auquel appartient le contour i;
  sinon
    pour *j allant de 0 à "nombre de contours adjacents au contour i"* **faire**
      si *zone de contact entre contour* i *et contour* j *inférieure à seuil A* **alors**
        les deux objets considérés **se touchent**;
      sinon si *zone de contact supérieure à seuil A et inférieure à seuil B* **alors**
        les deux objets considérés **se chevauchent**;
      sinon
        (si zone de contact supérieure à seuil B);
        si *contour* i *moins long que contour* j **alors**
          l'objet auquel appartient le contour i **recouvre** l'objet auquel appartient le contour j;
        sinon
          l'objet auquel appartient le contour i **est recouvert par** l'objet auquel appartient le contour j;

déterminée à l'issu du procédé décrit dans l'algorithme 1. Ceci doit donc intervenir à posteriori. Cette procédure est décrite dans l'algorithme 2.

Algorithme 2 : Complément des relations spatiales

Entrées : Liste des relations spatiales
Sorties : Liste des relations spatiales
pour *chaque couple d'objets (i,j)* **faire**
  si *il n'existe pas de relation entre* i *et* j *dans la liste des relations* **alors**
    rajouter une relation "i est disjoint de j" dans la liste;
    rajouter une relation "j est disjoint de i" dans la liste;

Il convient d'apporter une précision en ce qui concerne l'algorithme 1, et plus précisément au sujet des seuils mentionnés : La comparaison des différentes longueurs caractéristiques décrites dans la figure 3.8 avec des valeurs seuils est au cœur de notre méthode. Prenons l'exemple de la figure 3.9 : pour différentier les trois relations « Recouvre/est recouvert par », « chevauche » et « touche », nous considérons les longueurs :

- du contour de l'objet rouge ;
- du contour de l'objet bleu ;
- du contour commun aux deux objets.

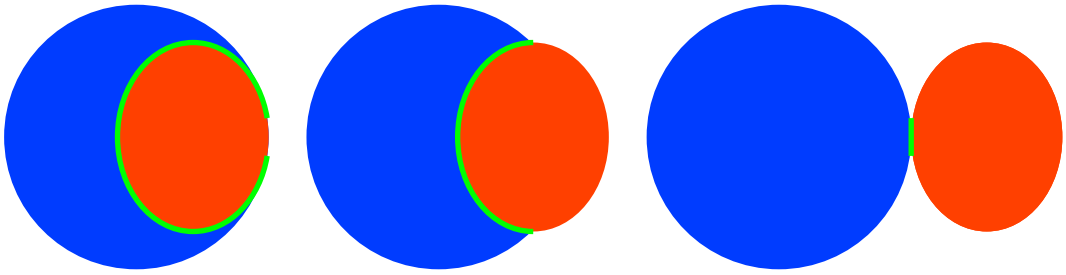


FIG. 3.9 – Différentiations des relations spatiales à l'aide d'un seuil

On peut prendre par exemple un seuil A égal à 10% et un seuil B égal à 90%. Dans ce cas, on déclarera que les deux objets *se touchent* si et seulement si la longueur de leur contour commun (en vert sur la figure) est inférieure à 10% de la longueur du contour de l'objet bleu **et** est inférieure à 10% de la longueur du contour de l'objet rouge. On pourra de même déclarer que l'objet rouge *recouvre* l'objet bleu si et seulement si la longueur du contour commun est supérieure à 90% de la longueur du contour de l'objet rouge. Enfin, les deux objets se chevauchent dans tous les autres cas.

Les seuils tels que décrits ici doivent être déterminés par expérimentation, afin de trouver quels seuils permettent d'obtenir l'interprétation la plus proche de celle d'un utilisateur. De plus, ces seuils, qui sont proportionnels aux longueurs des contours des objets peuvent être complétés par des seuils en distance absolue dans des cas limites. Un exemple de cas limite est proposé avec la figure 3.10. Sur cette figure, l'objet rond est trop petit par rapport à l'objet carré pour qu'on utilise un seuil proportionnel. En effet, les deux objets doivent être décrits comme se touchant, mais le contour commun mesure plus de dix pour cent de la taille de l'objet rond ce qui, avec le système des

seuils proportionnels, décrit une relation de chevauchement incorrecte. Le couplage de la méthode des seuils proportionnels avec des seuils absolus permet de pallier ce problème.

FIG. 3.10 – Cas limite nécessitant l'utilisation d'un seuil en valeur absolue

Par ailleurs, Nous devons prendre ici en compte une particularité de la relation *contient/est contenu par* : la transitivité. En effet, il s'est avéré que le jugement humain a tendance à accorder cette propriété au couple de relations de contenance. La figure 3.11 montre un exemple où les objets 1 et 3 seraient normalement décrétés *disjoints* d'après les caractéristiques que nous avons définies plus tôt, alors qu'ils sont en réalité perçus par un utilisateur comme reliés par une relation de contenance par transitivité.

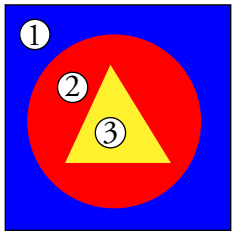


FIG. 3.11 – Transitivité de la relation de contenance

Il faut donc là aussi ajouter une procédure chargée de rechercher tous les cas où un objet contient un deuxième objet qui en contient un troisième pour mettre à jour la relation de contenance entre le premier et le troisième objet.

Chapitre 4

Implémentation

Ce chapitre a pour vocation de décrire les détails de l'implémentation de la méthodologie exposée dans le chapitre précédent. Nous exposons les problèmes qui se sont posés lors de l'implémentation de notre système et les éventuelles adaptations que nous avons dû effectuer par rapport à notre méthodologie de départ.

4.1 Détection des objets

Comme nous l'avons vu dans le chapitre 3, l'extraction des objets se fait en deux étapes : d'abord, la segmentation de l'image en zones de couleur uniforme et ensuite, la récupération des contours. Nous détaillons les implémentations de chacune d'entre elles ci-après.

4.1.1 Segmentation de l'image selon les couleurs

La segmentation est effectuée selon le nouvel espace des couleurs HCL développé par Sarifuddin et al. [26]. Le choix de cet espace se justifie par le fait qu'il est perceptuellement plus uniforme que la plupart des modèles de couleur proposés à date. Il convient donc parfaitement à notre utilisation.

Le principe est simple : on segmente l'image selon neuf teintes différentes :

- rouge
- vert
- blanc
- marron
- bleu
- gris
- jaune/orangé
- violet
- noir

Ces teintes sont fixées par des seuils sur chacun des trois axes définissant l'espace HCL. La technique de décomposition est restrictive dans le sens où elle ne permet pas de distinguer des variations de couleur au sein d'une même teinte, comme illustré dans l'exemple de la figure 4.1. Elle convient toutefois parfaitement pour notre application, le but n'étant pas ici de proposer un algorithme de reconnaissance d'objet performant, mais de disposer d'une base sur laquelle implémenter notre méthode de détection des relations spatiales. Il suffit simplement ensuite de tenir compte de cette limite lors de la mise au point des bancs de tests. De plus, notre système de DRS pourra toujours être plus tard couplé à un procédé de reconnaissance d'objet plus performant.

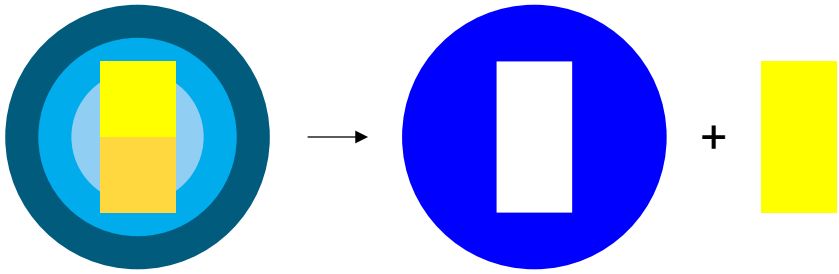


FIG. 4.1 – Limitation de la segmentation en couleurs

4.1.2 Récupération des contours

La récupération des contours ne pose pas vraiment de problème en soi. Les neuf images contenant chacune une plage de couleurs différente produites par la segmentation de l'image décrite ci-dessus sont traitées de la façon suivante :

Algorithme 3 : récupération des contours

Entrées : Image contenant une plage de couleur
Sorties : Image contenant les contours des zones de couleur
pour *i allant de 1 à nb de pixels de l'image* **faire**
  si *aucun des 4 pixels aux points cardinaux du pixel i n'est* vide **alors**
    mettre pixel n° i à vide;
  i ← i + 1;

Cela reprend le principe de la fenêtre posée sur chacun des pixels d'un objet, comme illustré dans la figure 4.2. Si le pixel est proche d'un région vide, c'est qu'il fait partie du contour de l'objet. Sinon, c'est qu'il se trouve à l'intérieur et on peut donc l'effacer.

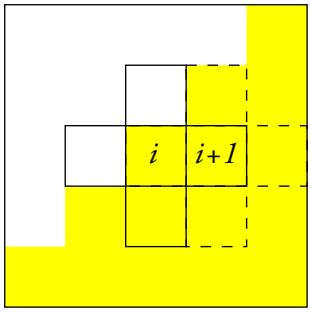


FIG. 4.2 – Récupération des contours

Dans le cas de la figure 4.2, le pixel n° i serait conservé, alors que le pixel n° $i+1$ serait effacé. Chacune des neuf images ne contient donc plus en sortie de cette procédure que des contours d'un pixel d'épaisseur. Il ne reste qu'à regrouper tous ces contours au sein d'une même image et à passer cette dernière image en paramètre pour la procédure de récupération des informations détaillée ci-après.

4.2 Récupération des informations utiles

Il s'agit dans cette section de récupérer de manière efficace les informations décrites dans la section 3.5.

4.2.1 Parcours systématique de l'image

Pour commencer, nous devons considérer que nous avons affaire à une image plane au sens où nous l'avons vu dans la section 3.2.2. Ceci signifie que nous devons parcourir l'image entièrement (pixel par pixel) pour détecter puis traiter chaque contour. L'algorithme est donc avant tout constitué d'une grande boucle parcourant systématiquement tous les pixels de l'image, en partant en haut à gauche et en finissant en bas à droite [1].

[1] Ceci n'est qu'une convention pour faire correspondre le comportement du programme avec celui d'un être humain, permettant ainsi de suivre la procédure plus facilement lors des tests. Toutefois, l'ordre d'étude des pixels et donc l'ordre de découverte des pixels a son importance au niveau de la détection d'un contour *intérieur* par exemple.

Pour chaque pixel étudié, plusieurs cas de figure se présentent :

Le pixel est « vide ». Dans ce cas, on l'ignore tout simplement, car un pixel vide est nécessairement situé dans la partie intérieure d'un objet, et nous avons vu dans le chapitre 3 que les informations pertinentes se situaient au niveau des contours. C'est d'ailleurs pour cette raison que les pixels non pertinents sont « vidés » lors de la récupération des contours des objets.

Le pixel n'est pas « vide ». Cela signifie que le pixel appartient au contour d'un objet, et c'est ce qui nous intéresse. Dans ce cas, deux cas de figure se présentent à nouveau :

1. Si le pixel n'est pas marqué, cela signifie que le contour est rencontré pour la première fois. On *suit* alors le contour auquel appartient ledit pixel (cette procédure sera détaillée ci-après). Le but est de noter les identifiants des contours adjacents à celui que l'on suit, pour pouvoir déterminer les objets qui sont en contact avec l'objet dont on suit le contour. On veut de plus déterminer la longueur de la partie commune entre le contour suivi et chacun des contours adjacents afin de différencier par la suite certaines relations, comme expliqué dans la partie 3.3
2. Si le pixel est marqué, cela signifie que le contour a déjà été suivi. Dans ce cas, on ignore le pixel.

Cette procédure permet aussi de déterminer si un contour est un contour *intérieur* ou *extérieur*, et c'est notamment ici que la notion d'ordre de parcours de l'image est importante. En effet, un moyen efficace de détecter si un contour est un contour intérieur ou extérieur est d'utiliser une **pile** et de raisonner sur l'ordre d'apparition des contours.

L'idée est de considérer chaque ligne du tableau représentant les pixels de l'image séparément. Lorsque l'on parcourt un par un tous les pixels d'une même ligne, on empile l'identifiant de chacun des pixels non vides rencontrés dans la pile. Ceci nous permet de connaître l'identifiant du pixel précédemment rencontré lorsque l'on étudie un pixel. Si l'identifiant du dernier pixel non vide rencontré est le même que celui du pixel étudié, alors on peut en déduire que le pixel étudié appartient à un contour *intérieur* de l'objet. La figure 4.3 donne une illustration de ce principe.

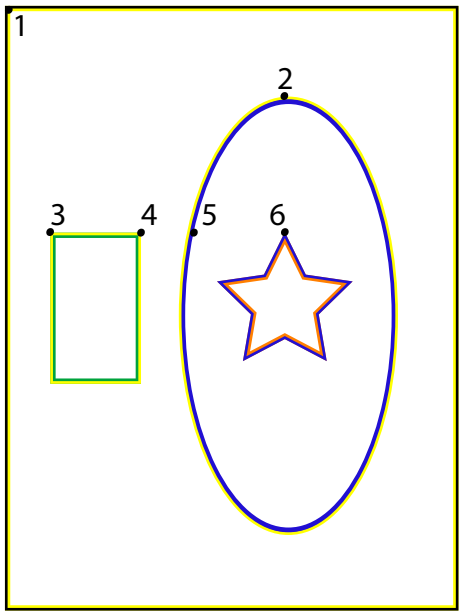


FIG. 4.3 – Utilisation de la pile pour la relation de contenance

Dans cette figure, le premier pixel rencontré est le pixel en ①. Il fait partie du contour jaune qui est donc suivi, et chaque pixel du contour est marqué comme ayant déjà été traité. Ensuite, tous les pixels que l'on rencontre jusqu'à atteindre ② sont ignorés, car ils sont soit vides, soit marqués dans le cas où ils appartiennent au contour jaune du fond de l'image. Jusque-là, la pile est inutile.

S'il l'on s'intéresse à la ligne à laquelle appartient le pixel en ②, on peut par contre tirer profit du principe énoncé ci-dessus. Étudions l'état de la pile tout au long de cette ligne :

- Au départ, la pile est vide[2] ;
- lorsqu'on étudie le premier pixel de la ligne, on découvre qu'il a déjà été suivi. On ignore donc ce pixel mais on empile tout de même son identifiant dans la pile. Elle contient alors : (Id_objet_jaune) ;
- on arrive ensuite en ② (tous les pixels entre ① et ② sont ignorés car ils sont vides ou déjà marqués). Le pixel en ② n'est pas marqué, il faut donc suivre le contour auquel il appartient. Par ailleurs, son identifiant est le même que celui que l'on a empilé (propriété résultant des caractéristiques du tableau récupéré en

[2]La pile est réinitialisée à chaque retour à la ligne.

sortie de la détection des objets). On en conclut donc qu'il s'agit d'un contour intérieur. En effet, ce pixel ne peut pas appartenir au même contour que celui auquel appartient le premier pixel de la ligne d'une part, car dans ce cas il aurait été marqué comme traité lors du suivi de ce contour et ne peut pas appartenir à un autre objet d'autre part, car il possède le même identifiant que celui du pixel en ①.

La même situation se retrouve en ③, mais l'étude du comportement de la pile dans la ligne complète à laquelle appartiennent ③, ④, ⑤ et ⑥ permet de mieux saisir la méthode :

- Pour commencer, encore une fois, la pile est vide.
- On empile ensuite l'identifiant du premier pixel. État de la pile : (Id_objet_jaune) ;
- En ③, on empile encore une fois (Id_objet_jaune), et on suit le contour. On déduit aussi que c'est un contour intérieur, de la même façon que pour le contour de l'élipse rencontré en ②. État de la pile : (Id_objet_jaune,Id_objet_jaune)
- pour chacun des pixels situés entre ③ et ④ inclusivement, on empile Id_objet_jaune, et on ignore le pixel. État de la pile en ④ : (Id_objet_jaune, . . . , Id_objet_jaune) ; ;
- en ⑤, on empile encore une fois Id_objet_jaune, lorsque l'on rencontre le contour jaune, et on ignore le pixel car le contour a déjà été suivi en ① donc le pixel est marqué. État de la pile identique + 1 "Id_objet_jaune" ;
- en ⑤ toujours, on rencontre ensuite le contour bleu. On doit donc empiler l'identifiant de l'objet bleu. On ignore encore une fois le pixel car le contour bleu a lui aussi été suivi en ①. État de la pile : (Id_objet_jaune, . . . , Id_objet_jaune, Id_objet_bleu)
- lorsqu'on arrive finalement en ⑥, le pixel détecté possède le même identifiant que celui qui est dans la pile et il n'est pas marqué, il appartient donc à un contour intérieur.

Cette technique a l'avantage de s'intégrer à la boucle principale de parcours d'image. Elle n'engendre donc pas d'itérations supplémentaires et n'impose que la création d'une pile peu coûteuse. De plus, elle permet à elle seule de déterminer le couple de relations *contient/est contenu dans* de manière certaine d'après les principes que nous avons énoncés dans la section 3.5. Elle constitue une amélioration considérable par rapport à la méthode que nous avions imaginée en premier lieu dans la section 3.3. En effet, le

principe consistant à dire qu'un objet en contient un autre lorsqu'un de ses contours n'est bordé que par un seul autre contour et que sa longueur est supérieure à celle de cet autre contour pose le problème suivant : tel qu'illustré dans la figure 4.4, dès qu'un objet contient au moins deux autres objets, elle devient inexploitable.

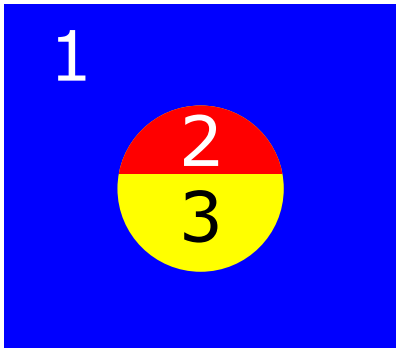


FIG. 4.4 – Exemple de cas où le suivi des contours ne suffit pas

Il est donc plus sûr et plus rapide d'utiliser une pile comme nous le proposons pour traiter le problème des contours intérieurs.

4.2.2 Suivi des contours

Le suivi des contours est une des deux étapes importantes des la méthode de DRS que nous avons mise au point, la seconde étant la déduction des relations spatiales d'après les informations extraites de l'image. Il s'agit ici justement d'extraire les informations pertinentes pour chaque contour cité dans la partie 3.3 :

1. La longueur du contour.
2. Les identifiants des objets en contact avec l'objet étudié.
3. La longueur de la zone de contact entre le contour et les différents contours avec lesquels il est en contact.

Le suivi des contours est appelé à chaque fois que l'on rencontre un *nouveau* pixel, c'est à dire un pixel non vide et non marqué. Le processus consiste à suivre les pixels ayant les mêmes identifiants, jusqu'à fermer le contour, et pour chaque pixel suivi, à le marquer comme tel, le compter, et compter les pixels adjacents pour chaque contour

en contact avec le contour étudié, en prenant garde à ne pas compter deux fois le même pixel, ni à en oublier. Toutes ces informations sont stockées dans la structure de données représentant le contour.

Pour illustrer la méthode du suivi des contours, nous allons utiliser la figure 4.5. Cette figure représente un état intermédiaire de la procédure. Nous sommes dans le cas où l'on suit un contour rouge qui est bordé par un contour bleu [3].

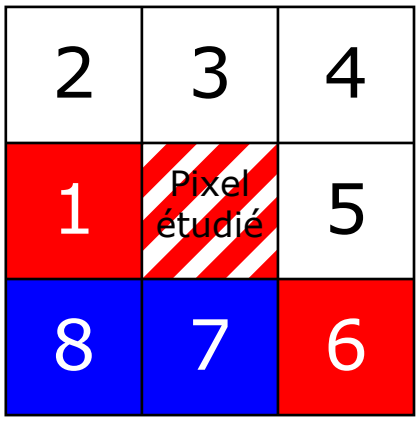


FIG. 4.5 – Suivi des contours

Chaque case du tableau représente un pixel de l'image. Il y en a donc neuf en tout. Le pixel en cours d'étude à ce stade de la procédure est celui situé au centre de l'image. On va considérer les huit pixels qui bordent le pixel en cours d'étude. Pour ce faire, on les étudie dans l'ordre indiqué par les numéros. On commence donc par le pixel au milieu et à gauche. Si l'on considère que le suivi a été fait de gauche à droite (le cas contraire est symétrique), alors c'est le pixel que l'on vient d'étudier, et il a été marqué comme tel à l'étape précedente. On l'ignore donc. Les pixels blancs représentent en fait ici des pixels vides, ils sont donc ignorés également. Lorsqu'on arrive sur le pixel n° 6 qui est rouge, il s'agit en fait de la suite du contour que l'on suit, on va donc le marquer comme tel, pour s'y rendre à l'étape suivante. On ne va toutefois pas s'y rendre immédiatement, il va d'abord falloir comptabiliser les autres pixels, en l'occurrence, les

[3] L'utilisation des couleurs ici n'est qu'une manière de rendre la figure plus lisible, on traite en réalité directement les identifiants des contours.

pixels bleus n° 7 et 8.

Il est à noter qu'il faut faire attention lors du comptage, car dans le cas de la figure 4.5 par exemple, les pixels bleus ont à priori déjà été comptabilisés lors de l'étape précédente, et lors de celle encore avant. Si le pixel 5 avait été rouge, et le 6 bleu, alors il aurait fallu incrémenter le nombre de pixels bleus de 1, correspondant au nouveau pixel détecté (le n° 6 dans le cas présent). On peut assimiler ce procédé à celui de la fenêtre coulissante.

Pour clore la description du suivi des contours, nous devons rapporter un problème gênant qui est apparu lors des premiers tests et qui a nécessité une adaptation de la méthode que nous venons d'énoncer : il s'agit des contours comportant des angles aigus.

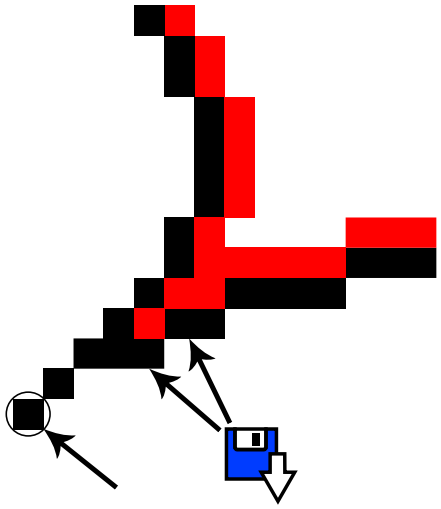


FIG. 4.6 – Problème lors du suivi des contours original

La figure 4.6 donne une idée du problème : lorsque l'on arrive au pixel entouré (pointé par la flèche en bas à gauche), on se retrouve dans une impasse, sans pour autant avoir fermé le contour. Ceci vient de la pointe induite lorsqu'un contour possède un angle très vif comme c'est le cas dans la figure. Une autre caractéristique de ce genre de contour qui va nous être utile ici est qu'il possède nécessairement un ou plusieurs pixels qui ont au moins *trois* voisins appartenant au même contour au lieu de *deux* comme

c'est le cas pour le reste du contour [4]. Pour pallier ce problème, nous introduisons donc une mémorisation de notre trajet qui fonctionne de la manière suivante :

Lorsque l'on rencontre un pixel ayant au moins trois voisins appartenant au même contour (tel qu'un de ceux pointés par les deux flèches indiquant la mémorisation sur la figure), nous ignorons toujours le pixel « d'où l'on vient », car il a déjà été traité, mais il reste toujours deux pixels possibles pour continuer le contour. Dans ce cas, nous en choisissons un qui sera étudié à la prochaine étape, et nous mémorisons la position de l'autre. Lorsque l'on arrive dans une impasse, il nous suffit de poursuivre le suivi des contours à la position ainsi mémorisée.

Cette méthode permet de résoudre entièrement le problème à partir du moment où l'on implémente la procédure de manière récursive.

4.3 Formulation des relations spatiales

4.3.1 Procédure principale

La procédure de détermination des relations spatiales reprend le principe décrit dans la section 3.6. Les relations sont traitées dans un ordre précis et déterminées par élimination. Quelques factorisations de critères de différentiation permettent tout de même de rendre l'arbre décisionnel plus efficace, et donc le procédé plus rapide. L'implémentation JAVA de l'algorithme final est disponible en annexe A.1.

4.3.2 Inférence

La procédure de détermination des relations spatiales au niveau local permet donc à elle seule de déterminer une partie des relations. Pour la compléter, on utilise ensuite une inférence basée sur la transitivité. Cette inférence se charge d'une part de compléter la formulation des relations spatiales en détectant les relations de disjonction et d'autre part de corriger certaines formulations pour les faire correspondre au jugement humain.

Les relations de disjonction sont traitées simplement : on étudie un par un tous les couples d'objets possibles et on vérifie s'il existe déjà une relation entre eux qui aurait

[4] un pixel est généralement en contact avec deux autres pixels appartenant au même contour : le pixel « d'où l'on vient » et celui « où l'on va ».

été déterminée lors de la procédure principale. Dans le cas où aucune relation n'a été formulée entre deux objets, c'est qu'ils n'ont pas de contour commun, et donc qu'ils sont disjoints.

Le traitement que nous proposons pour la formulation de la relation de disjonction détecte bien les relations de disjonction, mais en réalité, la relation en elle-même pose un problème, comme le montre la figure 3.11 de la section 3.6. En effet, si les objets n° 1 et n° 3 de cette figure sont techniquement disjoints au sens où nous l'avons défini, ils sont généralement plutôt perçus tels que l'objet 1 contient l'objet 3. Pour corriger cette particularité, il suffit de rajouter une procédure agissant directement sur la liste des relations spatiales produites lors de la procédure principale, puis complétée à l'aide des inférences pour les relations de disjonction. Cette procédure est présentée dans l'agorithme 4

Algorithme 4 : Transitivité de la contenance

Entrées : Liste des relations spatiales
Sorties : Liste des relations spatiales
pour *i allant de 1 à "taille de la liste des relations"* **faire**
  si *relation n° i = "contient"* **alors**
    mémoriser l'objet **A** et l'objet **B** concernés;
    pour *j allant de 1 à "taille de la liste des relations"* **faire**
      si *premier objet de la relation n° j = objet **B** **et** relation n° j = "contient"* **alors**
        mémoriser le second objet de la relation n° j en objet **C**;
    il faut maintenant mettre à jour la relation A - C :
    pour *k allant de 1 à "taille de la liste des relations"* **faire**
      si *premier objet de la relation n° k = objet **A** **et** second objet de la relation n° k = objet **C*** **alors**
        mettre à jour la relation n° k en "contient";
    terminer en mettant à jour la relation symétrique :
    pour *k allant de 1 à "taille de la liste des relations"* **faire**
      si *premier objet de la relation n° k = objet **C** **et** second objet de la relation n° k = objet **A*** **alors**
        mettre à jour la relation n° k en "est contenu dans";

On peut noter que cette procédure n'est pas récursive, c'est à dire que si la transitivité s'étend sur plus de trois objets, on ne met à priori à jour que la relation entre le premier et le troisième, mais on ne descend pas jusqu'au quatrième. Ce n'est en réalité pas le cas, et la procédure met bien a jour la relation entre tous les objets reliés par des relations de contenance par transitivité. Ceci est obtenu grâce à la construction particulière de la liste des relations. En effet, la liste est ordonnée selon l'ordre d'apparition dans l'image des objets. Ceci implique que les relations apparaissent dans la liste dans l'ordre suivant :

1 relation A $\subset$ B

2 relation A $\subset$ C

3 relation A $\subset$ D

4 relation B $\subset$ C

5 relation B $\subset$ D

6 relation C $\subset$ D

Imaginons que l'on ait une relation de contenance répartie de la manière suivante : (les relations notées XXX sont quelconques [5])

1er objet	2e objet	relation
A	B	contient
A	C	XXX
A	D	XXX
B	C	contient
B	D	XXX
C	D	contient

Ce tableau représente ce que l'on obtient typiquement après l'application de la procédure principale de formulation des relations spatiales et de la procédure de détection des relations de disjonction en complément. Clairement, on a ici un cas de relation de contenance A $\subset$ B $\subset$ C $\subset$ D

[5] Ce sont en réalité toutes des relations de disjonction, d'après la configuration requise pour que l'on ait affaire à une relation de contenance par transitivité. Toutefois, la relation qui existait au préalable n'importe pas car elle sera, quoiqu'il en soit, remplacée par une relation de contenance.

D'après la construction de la boucle et l'ordre de la liste, la première relation modifiée sera la relation A $\subset$ C, grâce aux relations A $\subset$ B et B $\subset$ C qui sont déjà connues. On obtient alors par transitivité :

1er objet	2^{e} objet	relation
A	B	contient
A	C	contient
A	D	XXX
B	C	contient
B	D	XXX
C	D	contient

Les relations A $\subset$ C et C $\subset$ D permettent ensuite de déterminer la relation A $\subset$ D, et on obtient :

1er objet	2^{e} objet	relation
A	B	contient
A	C	contient
A	D	contient
B	C	contient
B	D	XXX
C	D	contient

Pour finir, les relations B $\subset$ C et C $\subset$ D permettent d'obtenir la dernière relation qui se trouve être la relation B $\subset$ D.

1er objet	2^{e} objet	relation
A	B	contient
A	C	contient
A	D	contient
B	C	contient
B	D	contient
C	D	contient

La procédure permet donc un ajustement de **toutes** les relations de contenance par transitivité et est efficace grâce à la structure ordonnée de la liste des relations, car cela lui permet de se limiter à un seul passage.

4.4 Complexité spatiale

Nous nous intéressons ici à la complexité spatiale des informations stockées dans le cadre de notre méthode. En effet, ces informations sont vouées à être stockées dans une base de données, et il faut donc chercher à minimiser leur taille.

Considérons les structures de données impliquées ainsi que leurs composantes et voyons l'espace requis :

4.4.1 Les objets

Les structures de données représentant les objets contiennent trois informations :

- Un entier naturel supérieur ou égal à "1" et représentant l'identifiant de l'objet, codé sur 32 bits.
- Les informations caractérisant l'aspect de l'objet. Dans notre cas, il s'agit de la couleur, représentée par un entier là encore codé sur 32 bits, mais cette structure dépendra par la suite de l'utilisation faite de la méthode et de l'adaptation à d'autres processus de reconnaissance d'objets.
- La liste des contours composant l'objet. Nous avons choisi de représenter cette liste sous la forme d'un vecteur donc chaque composante pointe vers la structure de données caractérisant le contour concerné. Cette structure permet de simuler un tableau de taille variable et d'optimiser le stockage des informations.

4.4.2 Les contours

La structure de donnée représentant un contour est celle vers laquelle pointe le vecteur contenu dans un objet. Elle contient les informations suivantes :

1. Tout d'abord, l'identifiant de l'objet auquel appartient le contour, toujours représenté par un entier codé sur 32 bits. Cette information est redondante par rapport à celle stockée dans la structure de données représentant l'objet en question, mais est indispensable car le pointeur reliant les objets aux contours qui les composent est à sens unique, et sans cet identifiant au niveau des contours, on pourrait uniquement trouver les contours composant un objet, mais on ne pourrait pas trouver à quel objet appartient un contour de manière efficace.

2. Ensuite, la structure de données contient les caractéristiques de l'objet permettant la comparaison des contours entre eux. Dans notre cas, il s'agit encore de la couleur, donc un entier codé sur 32 bits. Ces caractéristiques sont nécessaires pour effectuer le suivi des contours.
3. Viennent les caractéristiques propres à chaque contour :
   - Sa longueur, entier codé sur 32 bits ;
   - Les longueurs des contours avec lesquels il est en contact, stockées dans un tableau d'entiers ;
   - Un booléen permettant de savoir si le contour est un contour intérieur ou extérieur à l'objet, nécessitant un bit supplémentaire.

4.4.3 Les relations

Les relations entre les objets doivent bien sûr elles aussi être stockées dans des structures de données performantes. C'est même en réalité la seule structure complètement dépendante de notre méthode, étant donné que les structures de données représentant les objets et les contours dépendent en partie du pré-processus de segmentation de l'image. Cette structure contient :

1. un pointeur vers le premier objet concerné par la relation ;
2. un pointeur vers le second objet concerné par la relation ;
3. une chaîne de caractères identifiant la relation.

Cette structure est minimale pour représenter une relation. La seule possibilité d'amélioration consiste par la suite à fusionner les trois structures que nous venons de voir afin d'économiser les pointeurs, et d'ajouter éventuellement de la compression. Mais cela implique de verrouiller le processus de reconnaissance des objets, ce qui est contraire à nos objectifs.

La structure générale que nous venons de décrire a donné de bons résultats en termes de complexité, permettant une détection des relations spatiales suffisamment rapide pour être exploitable, et nécessitant suffisamment peu d'espace mémoire pour pouvoir s'intégrer à une base de données.

En conclusion, nous avons finalement implémenté une procédure permettant de détecter les relations spatiales comme nous l'avions conçue au départ, à quelques aménagements près. La version implémentée est même plus performante que la version conçue à l'origine, grâce à la mise en place de structures performantes pour la liste des relations, les objets et les contours. Il reste à valider cette méthode par des tests et c'est ce qui est fait dans le chapitre suivant.

Chapitre 5

Experimentations et résultats

Nous allons décrire dans ce chapitre les différents tests que nous avons effectués avec notre méthode de détection des relations spatiales pour la valider, ainsi que les résultats que nous avons obtenus. Ces résultats seront analysés et nous verrons en quoi ils prouvent la robustesse et la précision de notre méthode et dans le cas de résultats erronés, dans quelle mesure ils peuvent contribuer à améliorer la précision de la méthode.

5.1 Élaboration des tests

La validation de notre approche s'est faite par la création de jeux de tests couvrant divers cas de figures de relations spatiales et par l'application des mesures de *rappel* et de *précision* communément utilisées pour l'estimation de l'efficacité d'un système de recherche d'information [24]. Dans la figure 5.1, le rappel est caractérisé par le quotient $\frac{\alpha}{\beta}$ et la précision par le quotient $\frac{\alpha}{\gamma}$. Une précision de 100% signifie donc que toutes les relations spatiales décelées par notre méthode le sont effectivement (c.-à-d., absence de fausses relations), alors qu'un rappel de 100% signifie que toutes les relations spatiales ont été identifiées par la méthode (c.-à-d. aucune omission de relations).

5.1.1 Mise en pratique

En pratique, il s'agit de constituer une collection d'images permettant de tester pour chaque type de relation spatiale si d'une part, elle est bien détectée dans divers

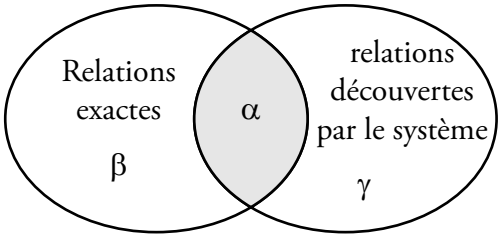


FIG. 5.1 – Calcul du rappel et de la précision

cas de figures et si d'autre part, il n'y a pas de relations erronées parmi celles détectées.. Par exemple, si l'on considère la relation « chevauche » dans la figure 5.2, on doit bien la déceler entre les objets 1 et 3, mais on ne doit la détecter ni entre les objets 1 et 2 (les objets 1 et 2 ne se chevauchent pas, ils sont disjoints), ni entre les objets 2 et 3 (les objets 2 et 3 se touchent, ils ne se chevauchent pas non plus).

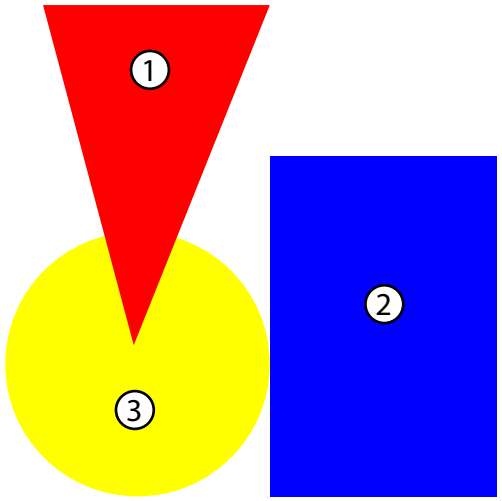


FIG. 5.2 – Exemple de test pour la relation « chevauche »

Pour cela, il nous faut donc créer un premier jeu d'images qui contient les configurations les plus diverses possibles mettant en jeu la relation étudiée, et un second jeu d'images testant cette fois les autres relations. Nous ne pouvons pas prétendre avoir testé **toutes** les configurations possibles pour chaque relation car il est impossible d'obtenir une liste exhaustive de toutes les combinaisons possibles, mais nous avons testé un

nombre suffisamment élevé de configurations pour chaque relation, ce qui nous permet de tester la fiabilité de la méthode pour les cas les plus communs.

5.1.2 Conditions d'expérimentation

Le problème qui se pose avec le protocole de test est que nous ne pouvons pas nous contenter de créer nous-mêmes l'ensemble d'images et prendre comme référence notre interprétation pour chacune d'entre elles. Pour obtenir des résultats objectifs, il fallait faire appel à des utilisateurs extérieurs au projet qui peuvent donner un avis neutre sur la qualité de la DRS que nous avons mise en place. Nous avons donc demandé à cinq intervenants extérieurs de participer non seulement à la création d'images de test mais aussi à l'analyse de ces images pour pouvoir comparer par la suite les différents résultats d'analyse.

Il est nécessaire d'avoir la participation de plusieurs personnes car les interprétations peuvent varier d'un individu à l'autre : là où certains voient deux objets reliés par la relation « se touchent », d'autres verront plutôt la relation « se chevauchent ». La figure 5.3 montre un exemple de cas ambigu entre ces deux relations.

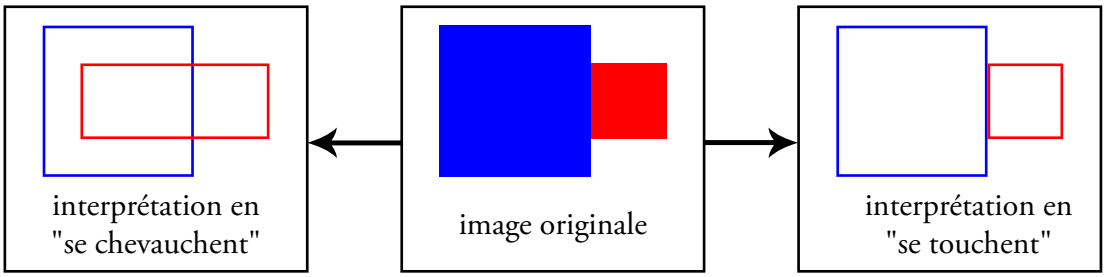


FIG. 5.3 – Deux interprétations possibles pour une même image

Les participants devaient donc produire une série d'une dizaine d'images chacun selon des critères détaillés dans la section suivante, puis nommer les relations présentes entre les objets dans ces images. Chaque participant devait par ailleurs analyser non seulement les images qu'il avait produites, mais aussi celles des autres.

Nous avons ensuite comparé les résultats donnés par l'application avec les interprétations de chacun des intervenants. Les résultats de la comparaison sont détaillés à la fin de ce chapitre.

5.2 Images de tests

Cette section donne un aperçu de la façon dont nous avons procédé pour créer nos jeux de test. En premier lieu, il convient de préciser que les images sur lesquelles nous travaillons dans cette section sont des images de *synthèse*. Nous ne pouvons en effet pas travailler sur des images « réelles » pour plusieurs raisons qui sont expliquées dans la section 5.2.1. Ensuite, la section 5.2.2 donne un exemple de jeu de test pour la relation « chevauche ». Cet exemple permet de mettre en évidence les problèmes liés à l'interprétation humaine ainsi que les décisions que nous avons dû prendre à ce sujet. La relation « chevauche » a été choisie car c'est elle qui permet d'illustrer le plus visiblement ce genre de problème d'interprétation.

5.2.1 Problèmes posés

Les problèmes que nous soulevons ici concernent uniquement la forme que doivent prendre les images de test.. Les images réelles ne sont pas utilisables pour tester notre méthode pour deux raisons principales :

– La première raison est la compression des images. Le codage JPEG en particulier, qui est le plus couramment utilisé pour les images réelles possède l'inconvénient de créer des zones « tampon » pour adoucir les contours en ajoutant une couleur intermédiaire entre les couleurs des deux objets concernés, comme illustré dans la figure 5.4.

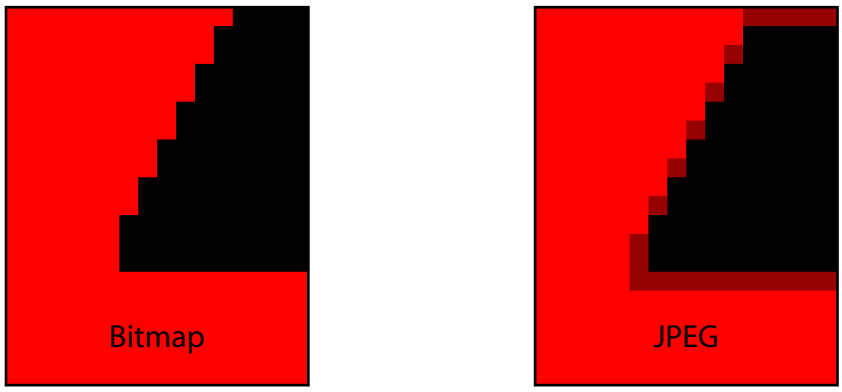


FIG. 5.4 – Problèmes liés à la compression des images

La figure représente deux agrandissements d'une même image, en bitmap à gauche et en JPEG à droite. Le problème de telles *zones tampon* est qu'elles créent un grand nombre d'objets parasites de taille négligeable (de l'ordre d'un pixel comme c'est le cas dans la figure 5.4) et peuvent fausser les résultats. Pour cette raison, nous n'utilisons que des images non compressées (Bitmap). Pour une utilisation sur des images « réelles » compressées en JPEG ou dans un autre format, il faudrait adapter la procédure de reconnaissance des objets ou à fortiori utiliser une méthode de segmentation différente.

– La seconde raison qui nous oblige à ne pas prendre d'images réelles est liée à la décomposition de l'image en objets. Si l'on considère l'image d'un simple ballon d'enfant par exemple, on obtiendrait avec les méthodes actuelles de décomposition (pas seulement celle que nous avons utilisée, mais la plupart des méthodes d'extraction d'objets existantes à date) le résultat présenté sur la figure 5.5.

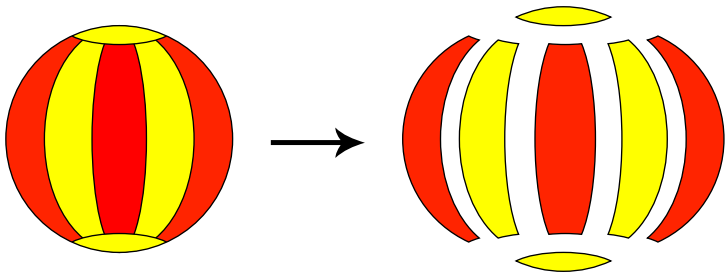


FIG. 5.5 – Problèmes liés à la décomposition des images en objets

On comprend facilement ce que cela peut donner pour une image plus complexe. Les objets sont difficilement descriptibles par un utilisateur, et le résultat d'un test utilisant une image réelle serait difficile à interpréter. La figure 5.6 représentant un dessin d'enfant ainsi que sa décomposition en objets illustre bien le problème à grande échelle : elle contient 79 objets à elle seule avec notre mode de décomposition. Le nombre de relations s'élève donc à $79^2 - 79 = 6162$ relations. Ceci n'est manifestement pas adapté pour des tests sur les relations spatiales étant donné qu'il faut vérifier les RS manuellement une par une.

Pour ces deux raisons, nous préférons synthétiser des images ne comportant que peu d'objets facilement identifiables (par exemple, des formes géométriques de base).

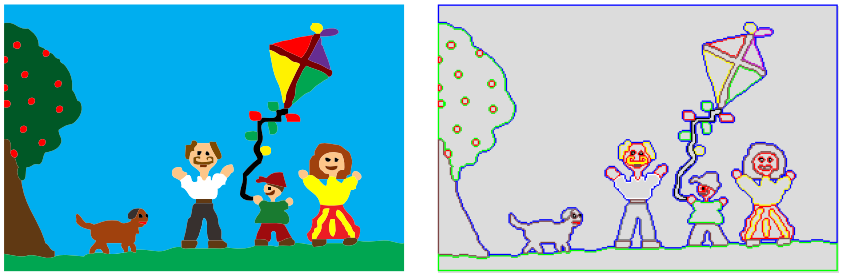


FIG. 5.6 – Illustration de la difficulté à décrire une image complexe

5.2.2 Exemple de jeu de test

Le jeu de tests que nous présentons concerne la relation « chevauche ». Nous avons tout d'abord un ensemble d'images permettant de tester le rappel pour cette relation (c'est-à-dire que toutes les relations de chevauchement sont bien détectées) :

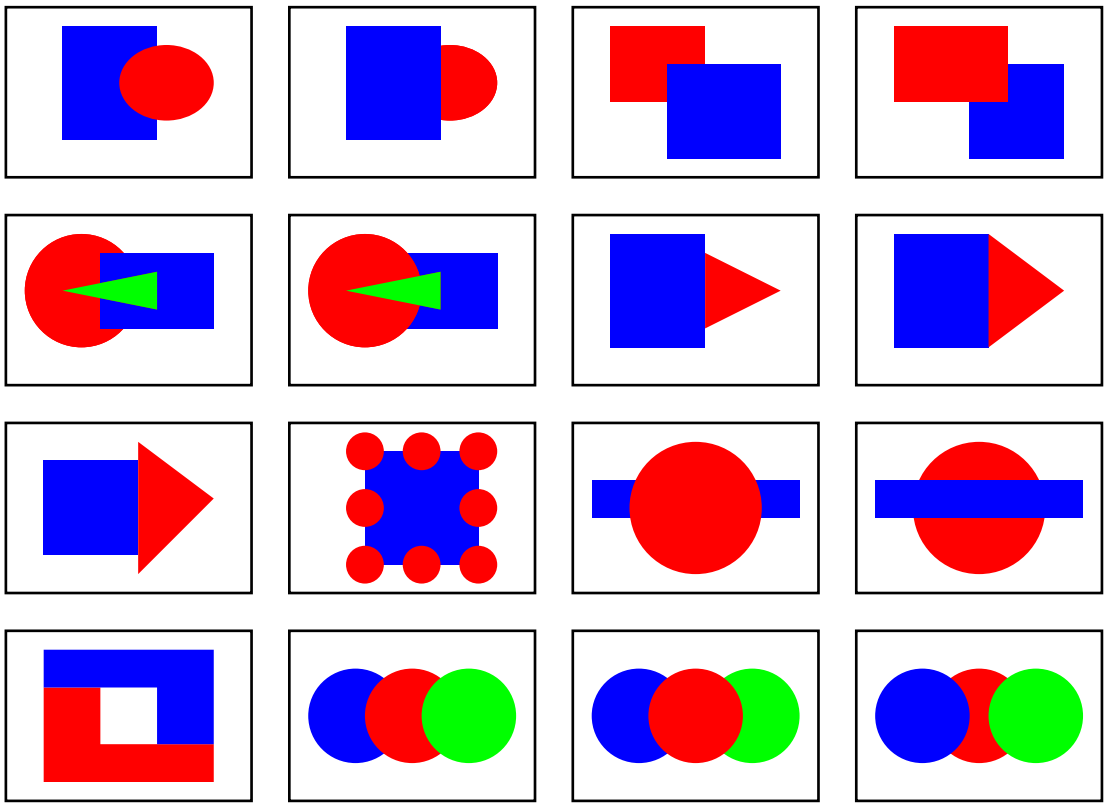


FIG. 5.7 – Exemples d'images de test de rappel pour la relation « chevauche »

La figure 5.7 montre un échantillon d'images utilisées pour les tests de rappel de la relation « chevauche ». Tous les objets de ces images (excepté le fond) sont reliés par une relation de chevauchement.

Nous avons ensuite un ensemble d'images permettant de tester la précision de la relation :

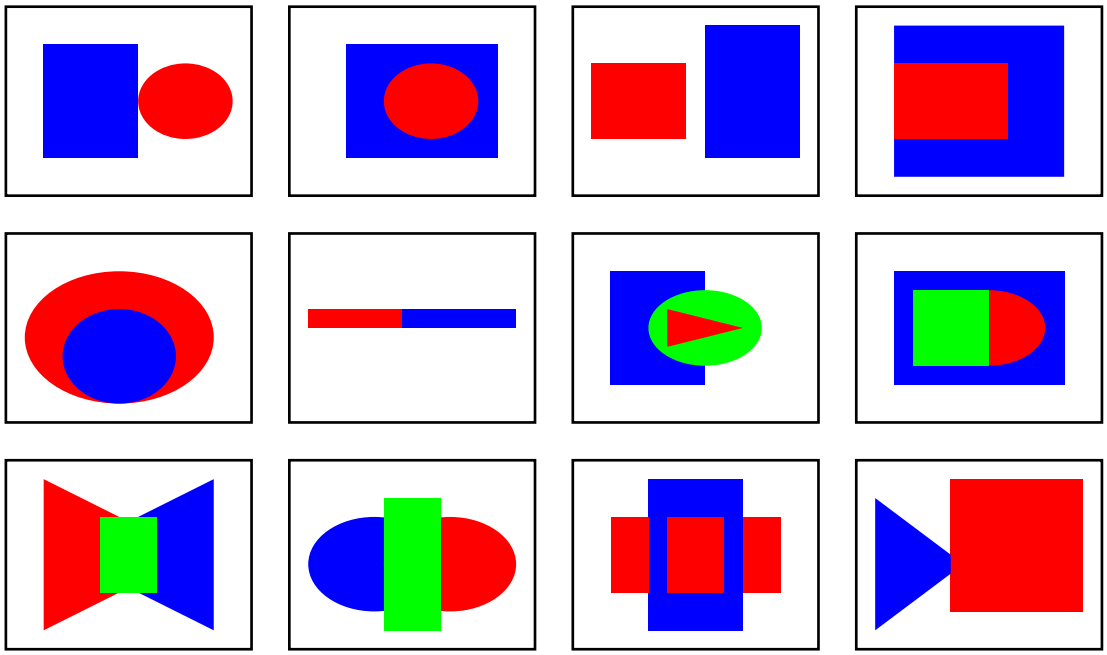


FIG. 5.8 – Exemples d'images de test de précision pour la relation « chevauche »

Dans les images présentées dans la figure 5.8, nous nous intéressons à la relation entre l'objet rouge et l'objet bleu. Dans aucun des cas il ne s'agit d'une relation de chevauchement, la méthode ne doit donc en théorie pas détecter de relation de chevauchement lors de l'analyse de ces images (exception faite pour l'avant-dernière image où les deux objets rouges sur les côtés chevauchent bien l'objet bleu au centre, mais c'est dans ce cas précis l'objet rouge *contenu* dans l'objet bleu qui nous intéresse).

Nous avons présenté dans cette section *deux* jeux de tests différents pour une raison de clarté, mais la plupart des images de ces jeux de tests peuvent en réalité permettre de tester les deux aspects (rappel et précision) pour plusieurs relations différentes. On pourrait donc concaténer le tout en un seul jeu de test.

5.3 Résultats obtenus et interprétation

Cette dernière section nous permet de décrire les résultats que nous avons obtenus à l'aide des tests décrits dans les sections précédentes et de les commenter.

Tout d'abord, le tableau 5.1 présente les résultats des tests effectués sur l'ensemble des images, classés par relation.

Relation	occurrences	occurrences non détectées	détections fausses
Disjoint	60	0	0
Touche	46	**2**	**4**
Chevauche	50	**6**	**8**
Recouvre	16	**2**	0
Est recouvert par	16	**2**	0
Contient	36	0	0
Est contenu dans	36	0	0

TAB. 5.1 – Résultats des tests classés par relation

On peut faire plusieurs remarques concernant ces résultats :

Tout d'abord, il y a des écarts importants au niveau du nombre d'occurrences (16 occurrences pour la relation « Recouvre » contre 60 pour la relation « Disjoint » par exemple). Ceci vient du fait que d'une part les relations *disjoint*, *touche* et *chevauche* sont réflexives alors que les trois dernières relations ne le sont pas (en conséquence de quoi le nombre d'occurrence est doublé pour les premières relations par rapport aux autres) et d'autre part parce qu'il existe plus de configurations possibles permettant d'obtenir la relation « disjoint » que de configurations permettant d'obtenir la relation « recouvre » par exemple.

Ensuite, La relation qui pose manifestement problème est la relation « Chevauche ». En effet, les valeurs présentées dans ce tableau sont corrélées et les quatre relations « touche » qui sont détectées en trop se retrouvent parmi les six relations « chevauche » non détectées. Le problème vient de la définition même de ces relations présentée en section 3.6. La prise en compte des seuls caractéristiques locales au niveau des contours et l'utilisation d'un seuil sur la longueur de contact entre deux objets pour déterminer la différence entre la relation « touche » et « chevauche » ne garantit pas des résultats justes dans certains cas limites.

Les cas limites dont nous parlons sont des cas où même un utilisateur aurait du mal à trancher. L'exemple de la figure 5.9 présente un tel cas de détermination difficile :

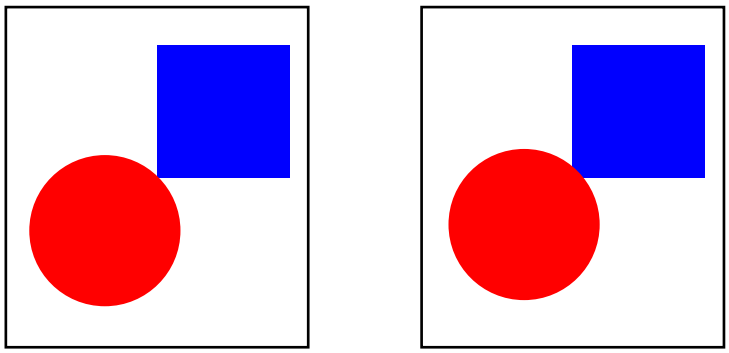


FIG. 5.9 – Cas de détermination non triviale

D'après les définitions théoriques que nous avons citées dans la section 3.2.1, les deux objets de l'image de gauche se *touchent* tandis que ceux de l'image de droite se *chevauchent*. En pratique, notre méthode détecte dans les deux cas la relation « se touchent ». Ce résultat fait donc partie des résultats erronés présentés dans le tableau 5.1. Toutefois, nous avons aussi pu constater que trois des cinq participants au test ont fait la même analyse que l'application et identifié dans les deux images la relation « se touchent ». Nous avons de plus remarqué par la suite lors de tests supplémentaires que le résultat de l'analyse des participants pouvait dépendre du contexte, et plus particulièrement de l'ordre dans lequel les images étaient présentées : dans le cas de deux participants, si la configuration de droite était présentée AVANT la configuration de gauche, alors les deux objets de l'image de droite étaient considérés comme *se touchant*, mais si la configuration de droite était présentée APRÈS la configuration de droite, les deux objets de l'image de droite étaient considérés comme « se chevauchant ». Les résultats erronés ne sont donc pas obligatoirement gênants suivant l'application que l'on veut faire de la DRS.

Le chapitre suivant va permettre de détailler le conclusions auxquelles nous sommes arrivés à ce sujet. Nous proposerons de plus dans ce chapitre des directions de recherche pour améliorer notre système.

5.4 Leçons apprises et propositions d'ameliorations

5.4.1 Leçons apprises

La détection des relations spatiales entre les objets présents dans une image est loin d'être une chose aisée. Elle fait appel à des critères de jugement propres à chaque individu et cela pose de problème lorsque l'on veut simuler le jugement humain à l'aide d'une procédure automatique. La première chose que nous avons apprise est donc qu'il n'existe pas *à priori* de définition universelle des relations spatiales et que l'on devra toujours bien préciser la nature de celles que l'on traite dans un système sous peine d'incompréhension de l'utilisateur.

En second lieu, l'approche que nous avons proposée pose les bases pour la détection de relations spatiales en décrivant une structure de traitement (récupération d'informations + analyse de ces informations) sur laquelle on peut sans difficulté élaborer un système plus complexe. Étant donnée que l'approche proposée pose les bases pour un système de DRS, il reste encore beaucoup à faire avant de pouvoir l'exploiter dans un système opérationnel de recherche d'image basée sur le contenu. Il faut en particulier raffiner les critères de différenciation des relations spatiales pour ne plus avoir de transition confuse entre certaines relations (les relations « chevauche » et « touche » comme indiqué dans la section 5.3 par exemple).

Enfin, ce qui nous a intéressé est la procédure de DRS elle même, la procédure de segmentation de l'image que nous avons utilisée reste limitée et il serait intéressant de coupler notre approche avec un système de reconnaissance d'objets plus élaboré, tel que celui de Grosky et Mehrota [13] ou bien celui de Mehrotra et Gary [20].

5.4.2 Propositions d'amélioration et travaux futurs

La prochaine étape pour le développement de notre approche serait donc le couplage avec un système de reconnaissance d'objets plus performant et par conséquent, l'adaptation à des formes complexes telles que définies dans le chapitre 2. Cette adaptation passerait certainement par des changements au niveau des caractéristiques décrivant chaque relation, et des ajouts de critères pour compléter la détection existante.

Pour ce qui est de l'affinage des critères de différenciation des RS justement, nous sommes arrivés à la conclusion que l'étude des objets au niveau local ne permettait pas de fixer des critères tout a fait satisfaisants. Il serait parfois souhaitable d'avoir une vue de l'objet dans son ensemble. Pour se faire, l'utilisation des plus petits rectangles conteneurs utilisés par Kim et Um [17] pourrait être avantageuse. Le problème posé par l'utilisation de ces rectangles conteneurs est qu'ils ne sont pas adaptés lorsque l'alignement des deux objets entre lesquels on essaye de déterminer une relation n'est pas proche de l'horizontale ou de la verticale. La figure 5.10 donne un exemple de cas où les plus petits rectangles conteneurs ne sont pas exploitables.

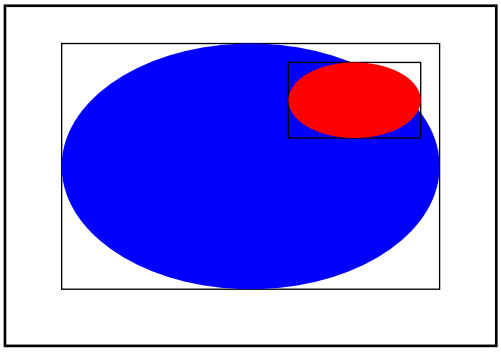


FIG. 5.10 – Exemple de cas où les rectangles conteneurs ne sont pas exploitables

Pour pouvoir tirer profit des plus petits rectangles conteneur, il faudrait tourner l'image jusqu'à aligner les objets de façon optimale, comme illustré par la figure 5.11

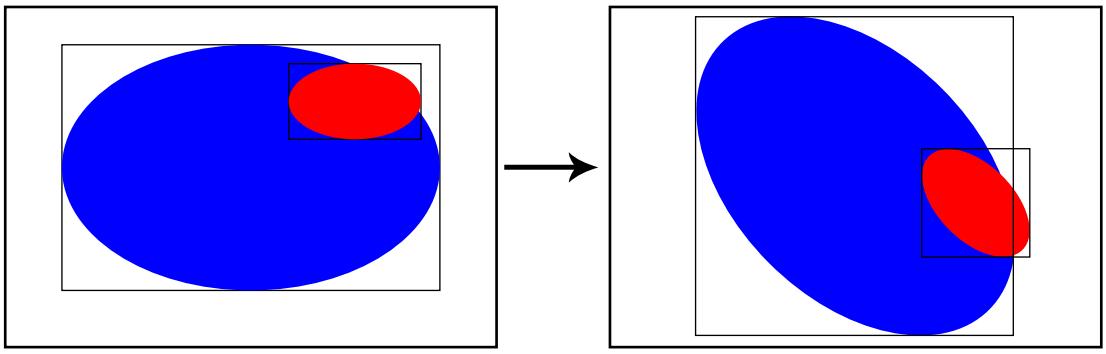


FIG. 5.11 – Utilisation optimale des plus petits rectangles conteneurs

L'autre solution est bien entendu de tourner les rectangles plutôt que l'image entière. Le problème d'un tel procédé est qu'il est très coûteux car il faut répéter la rotation de l'image ou des rectangles conteneurs pour chaque nouvelle paire d'objets étudiée et il est de plus difficile à concevoir (comment définir précisément ce qu'on appelle une configuration « optimale » pour une paire d'objets donnée ?). Ce pourrait être le sujet d'un projet de recherche complet et l'intégration d'une technique basée sur cette idée à notre méthode d'analyse locale au niveau des contours donnerait sans nul doute de meilleurs résultats et permettrait par exemple la clarification de la frontière entre les relations « touche » et « chevauche ». Le cas exposé dans la figure 5.9 pourrait alors être reclassé en relation « se touche » conformément aux définitions théoriques présentées dans la section 3.2.1.

Conclusion

Ce mémoire présente une technique de raffinement de la recherche de relations spatiales entre les objets situés dans une image par la prise en compte des relations spatiales entre les objets présents dans une image. La méthode présentée utilise des techniques de détection de couleurs et de contours pour reconnaître les objets présents dans une image, ainsi qu'un processus de suivi des contours pour déterminer les relations qui existent entre ces objets.

Le système que nous avons conçu est autonome et ne requiert d'action de l'utilisateur ni pour la reconnaissance des objets, ni pour la détection des relations spatiales. Il détecte de plus les sept relations primaires que l'on peut rencontrer dans une image plane avec un taux d'erreur tout à fait acceptable. Les améliorations proposées dans le dernier chapitre de ce rapport pourront permettre d'utiliser notre approche comme base d'un système de DRS qui, implanté dans un processus de recherche d'images basée sur le contenu, améliorera à terme la pertinence des résultats.

Annexe A

Algorithmes

A.1 Implémentation Java de l'algorithme de détermination des RS

```
private void detectionRelations(Vector relations, SimpleObject object,
    Vector edges, SimpleObject[] objects) {
  Vector objectEdges = object.edges();
  int j = 0;
  while (j < objectEdges.size()) {
    SimpleEdge edge = (SimpleEdge)objectEdges.get(j);
    for (int i = 1; i < nbObjects; i++) { // commence à 1 car le
       premier chiffre n'est pas représentatif d'un id, les
       identifiants commencent à 1
      if (i != object.id()) {
        SimpleRelation relation;

        int k = 0;
        while(((SimpleEdge)edges.get(k)).id() != i) k++;
        SimpleEdge edge2 = (SimpleEdge)edges.elementAt(k);

        if (edge.ids()[i] != 0) {
          if (edge.interior())
          relation = new SimpleRelation(object, objects[edge2.id()], "
             contient");
          else {
            boolean contained = false;
            int l = 0;
            SimpleRelation testRelation;
            while (l < relations.size()) {
              testRelation = (SimpleRelation)relations.get(l);
              if (testRelation.firstObject().id() == i
               testRelation.secondObject().id() == object.id()
```

```
                testRelation.relation().equalsIgnoreCase("contient")) {
                 contained = true;
                 break;
               }
             l++;
             }

             if (contained)
             relation = new SimpleRelation(object, objects[edge2.id()],
                 "est contenu par");
             else {
               int length1 = edge.length();
               int length2 = edge2.length();
               int contactlength1 = edge.ids()[i];
               int contactlength2 = edge2.ids()[edge.id()];

               if (contactlength1 >= (10*length1/100)  contactlength1 <
                   (90*length1/100)) {
                 if (contactlength2 >= (90*length2/100))
                 relation = new SimpleRelation(object, objects[edge2.id
                    ()], "est recouvert par");
                 else if (contactlength1 < contactlength2)
                 relation = new SimpleRelation(object, objects[edge2.id
                    ()], "est chevauché par");
                 else if (contactlength1 > 10)
                 relation = new SimpleRelation(object, objects[edge2.id
                    ()], "chevauche");
                 else relation = new SimpleRelation(object, objects[
                    edge2.id()], "touche");
               }
               else if (contactlength1 >= (90*length1/100))
                 relation = new SimpleRelation(object, objects[edge2.id
                    ()], "recouvre");
               else if (contactlength2 > 10*length2/100  contactlength2
                   > 10)
                 relation = new SimpleRelation(object, objects[edge2.id
                    ()], "est chevauché par");
               else relation = new SimpleRelation(object, objects[edge2
                  .id()], "touche");
             }
           }
           relations.add(relation);
         }
       }
     }
     j ++;
   }
 }
```

A.2 Parcours systématique de l'image

```
private Vector parcoursImage(int[][] idTab) {
  globalCheckTab = new boolean[h][w];
  for (int i = 0; i < h; i++) Arrays.fill(globalCheckTab[i], false);

  Stack stack = new Stack();

  int pixelColor = 0;
  int edgeCount = 0;
  int objectId = 0;
  int edgeId = 0;
  boolean interior = false;
  Vector edges = new Vector(50, 1000);

  for (int i = 0; i < h; i++) {
    for (int j = 0; j < w; j++) {
      pixelColor = colorTab[i][j];
      if (pixelColor != -1) {
        if (!globalCheckTab[i][j]) {
          interior = false;
          if (stack.isEmpty()) objectId = ++edgeCount;
            else {
            if ((((String)stack.lastElement()).split(";"))[0].equals("
               "+pixelColor)) {
              interior = true;
              objectId = new Integer((((String)stack.lastElement()).
                 split(";"))[1]).intValue();
            }
            else objectId = ++edgeCount;
          }
          edges.add(edgeId, suiviContour(i, j, objectId, interior));
          edgeId ++;
        }
        else stack.push(""+pixelColor+";"+idTab[i][j]);
      }
    }
    stack.clear();
  }
  return edges;
}
```

Bibliographie

[1] Chang, J.-W., Kim, Y.-J., and Chang, K.-J. A spatial match representation scheme for indexing and querying in iconic image databases. In *Proceedings of the sixth international conference on Information and knowledge management* (January 1997), ACM Press, pp. 169–176.

[2] Chang, S. K., SHI, Q. Y., and W.YAN, C. Iconic indexing by 2-d strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 3 (May 1987), 413–428.

[3] Deb, S., and Zhang, Y. An overview of content-based image retrieval techniques. In *Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)* (March 2004), vol. 1, IEEE Computer Society, pp. 59–64.

[4] Egenhofer, M., Frank, A., and Jackson, J. A topological data model for spatial databases. In *Proceeding of the first Symposium on the Design and Implementation of Large Spatial Databases* (1989), Springer-Verlag, pp. 271–286.

[5] Egenhofer, M., and J. Sharma, D. M. A critical comparison of the 4-intersection and 9-intersection models for spatial relations : Formal analysis, October 1993.

[6] Egenhofer, M. J. A formal definition of binary topological relationships. In *Proceedings of the 3rd Int. Conf. on Foundations of Data Organization and Algorithms* (1989), vol. LNCS 367, Springer-Verlag, pp. 457–472.

[7] Egenhofer, M. J. A model for detailed binary topological relationships. *Geomatica 47*, 3-4 (1993), 261–273.

[8] Egenhofer, M. J., and David M. Mark, J. H. The 9-intersection : Formalism and its uses for natural-language spatial predicates. Tech. Rep. 94-1.

[9] Egenhofer, M. J., and Herring, J. Categorizing binary topological relations between regions, lines, and points in geographic databases. `http://www.spatial.maine.edu/~max/9intReport.pdf`, 1990.

[10] El-Kwae, E. A., and Kabuka, M. R. A robust framework for content-based retrieval by spatial similarity in image databases. *ACM Transactions on Information Systems (TOIS) 17*, 2 (April 1999), 174–198.

[11] Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovi, D., and Equitz, W. Efficient and effective querying by image content. *Journal of Intelligent Information Systems 3*, 3–4 (July 1994), 231–262.

[12] Gao, H., Siu, W.-C., , and Hou, C.-H. Improved techniques for automatic image segmentation. *IEEE Transactions on Circuits and Systems for Video Technology 11*, 12 (December 2001), 1273–1280.

[13] Grosky, W. I., and R, M. Index-based object recognition in pictorial data management. *Computer Vision, Graphics, and Image Processing 42* (December 1990), 416–436.

[14] Gudivada, V. N. Or-string : A geometry-based representation for efficient and effective retrieval of images by spatial similarity. *Tech. Rep. CS-95-02.* (1995).

[15] Gudivada, V. N., and Raghavan, V. V. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems (TOIS) 13* (April 1995), 115–144.

[16] Hède, P., Moëllic, P.-A., Bourgeoys, J., Joint, M., and Thomas, C. Automatic generation of natural language descriptions for images. In *Proceedings of the RIAO-2004 Conference (Recherche d'Information Assistée par Ordinateur)* (April 2004).

[17] Kim, B., and Um, K. 2d+ string : A spatial metadata to reason topological and directional relationships. In *Proceedings of the 11th International Conference on Scientific and Statistical Database Management* (July 1999), IEEE Computer Society, p. 112.

[18] Kim, Y.-J., Sim, C.-B., and Chang, J.-W. Spatial match representation scheme supporting ranking in iconic images databases. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management* (November 1999), ACM, pp. 450–457.

[19] Lee, J., Oh, J., and Hwang, S. Strg-index : spatio-temporal region graph indexing for large video databases. In *SIGMOD '05 : Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2005), ACM Press, pp. 718–729.

[20] Mehrotra, R., and Gary, J. E. Feature-based retrieval of similar shapes. In *Proceedings of the Ninth International Conference On Data Engineering* (April 1993), p. 108.

[21] Missaoui, R., Sarifuddin, M., and Vaillancourt, J. An effective approach towards content-based image retrieval. In *Proceedings of the International Conference on Image and Video Retrieval (CIVR)* (July 2004), pp. 335–343. Dublin, Ireland.

[22] Missaoui, R., Sarifuddin, M., and Vaillancourt, J. Similarity measures for an efficient content-based image retrieval. In *IEEE Vision, Image & Signal Processing* (December 2005), vol. 152, pp. 875–887.

[23] Nabil, M., Ngu, A. H. H., and J.Shepherd. Picture similarity retrieval using the 2d projection interval representation. *IEEE Transactions on Knowledge and Data Engineering 8*, 4 (1996), 533–539.

[24] Salton, G., and McGill, M. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[25] Samet, H. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[26] Sarifuddin, M., and Missaoui, R. A new perceptually uniform color space with associated color similarity measure for content-based image and video retrieval. *ACM SIGIR Workshop on Multimedia Information Retrieval* (August 2005). Salvador, Brazil.

[27] Schneider, M., and Behr, T. Topological relationships between complex spatial objects. `http://www.cise.ufl.edu/tech_reports/tr04/tr04-011.pdf`, 2004.

[28] Smeulders, A. W. M., Worring, M., Santini, S., and A. Gupta, R. J. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Inteligence (PAMI) 22*, 12 (2000), 1349–1380.

[29] Traina, A. J. M., Balan, A. G. R., Bortolotti, L. M., and Jr., C. T. Content-based image retrieval using approximate shape of objects. In *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS'04)* (June 2004), IEEE Computer Society, pp. 91–96.

[30] Walker, A. R., Pham, B., and Moody, M. Spatial bayesian learning algorithms for geographic information retrieval. In *GIS '05 : Proceedings of the 13th annual ACM international workshop on Geographic information systems* (New York, NY, USA, 2005), ACM Press, pp. 105–114.