Partie B

Images similaires

Le terme d'objet de grande taille est naturellement associé aux images. Les images occupent, en effet, beaucoup d'espace, notamment dans le domaine de l'imagerie médicale [KSF+96].

Cette partie présente le résultat de la collaboration internationale (accords CNRS-CONICIT 5485 et 7202) avec Marta Rukoz, Professeur au Centre d'Informatique Parallèle et Distribuée (CCPD) de l'Université Centrale du Venezuela, à Caracas.

Cette partie est organisée comme suit. Le chapitre 3 décrit une application de biologie dans laquelle les images sont organisées à l'aide d'une structure particulière, l'arbre quaternaire. Cette structure est présentée en détail dans le chapitre 4. Ce même chapitre présente plusieurs approches appliquant les mécanismes de chevauchement, présentés dans le chapitre 1, aux images organisées en arbre quaternaire. Le chapitre 5 définit l'*Arbre Quaternaire Générique*, structure que nous proposons pour répondre aux problèmes posés par les utilisateurs de l'application présentée dans le chapitre 3. L'arbre Quaternaire Générique est comparé aux autres approches dans le chapitre 6.

Chapitre 3

Contexte : une application de traitement d'images

Résumé

Ce chapitre présente les besoins des utilisateurs d'une application de traitement d'images. Ces utilisateurs, des biologistes, appliquent, de manière coopérative, différents traitements sur différentes images originales. Ils souhaitent, pour optimiser leur application, disposer de mécanismes qui permettent d'organiser et de stocker leurs images originales, les images résultant d'opérations, et également les images intermédiaires produites par les traitements. Le traitement d'images est un domaine complexe, impliquant un processus d'essai-erreur et la nécessité de travailler parfois sur des petites régions d'image. En effet, les images médicales sont telles qu'un traitement peut améliorer certaines régions mais en bruiter d'autres, c'est-à-dire les rendre moins visibles. L'objectif final de l'application est d'offrir à chaque utilisateur la possibilité de construire des images où tous les éléments dignes d'intérêt pour l'utilisateur (ici des cellules) sont visibles, les régions de l'image ayant pu être traitées différemment.

Introduction

Ce chapitre présente l'application d'imagerie médicale dans le cadre de laquelle la structure d'*Arbre Quaternaire Générique*, présentée dans le chapitre 5, a été proposée. Cette application d'imagerie médicale fait partie du projet SIMA [1] [LR95, LRR98] de parallélisation d'algorithmes de traitement d'images.

L'objectif des traitements appliqués aux images est d'améliorer la qualité des images, selon des critères propres aux biologistes, utilisateurs de l'application. Ces améliorations consistent par exemple à faire apparaître plus clairement certains éléments de l'image, à montrer certaines caractéristiques ou à faire apparaître plus clairement des différences.

Exemple 3.1 La figure 3.1 présente une image avant et après traitement, le traitement consistant à faire apparaître, dans la zone située en haut à droite (les régions 0110 et 0111), des éléments peu visibles dans l'image initiale. Les régions 0110 et 0111 ont été assombries afin de rendre les cellules plus visibles. La figure 3.3 présente la différence des deux images de la figure 3.2 dont celle de droite correspond à une dilatation des régions 012 et 013 par rapport à l'image originale située à gauche de la figure. La dilatation joue le rôle d'une lime à épaissir en reconstruisant les bordures irrégulières [SM93] ; cette opération permet de fermer des contours des objets de l'image. Dans la figure 3.4, plusieurs traitements (dilatation, complémentaire, détection des bords, etc.) ont été appliqués afin de permettre la visualisation (voir figure 3.5) d'éléments peu visibles à l'origine dans l'image initiale. Ces images médicales ont été créées à partir de copies d'écran du prototype développé à l'Université Centrale du Venezuela [VR98, JMR+99b] (voir chapitre 5).

Afin de pouvoir appliquer ces traitements, les biologistes travaillent selon un processus d'essai-erreur, utilisant des opérations qui génèrent à chaque fois une nouvelle image résultat. Ces opérations peuvent être appliquées sur l'image entière ou sur des parties d'images (voir figures 3.1 et 3.2). En effet, une opération, appliquée à une image entière, peut améliorer certaines parties de l'image et en détériorer d'autres. Par conséquent, il est préférable d'appliquer localement certains traitements sur des parties de l'image et de recomposer l'image à partir de morceaux différemment traités. De plus, il est commun dans ce genre d'application de faire des modifications manuelles sur l'image (par exemple, re-dessiner un contour effacé par un traitement).

En définitive, de nombreuses images sont créées lors du processus d'amélioration d'une image. Au cours du traitement, les images résultats ou intermédiaires doivent être sauvegardées, afin de permettre des retours en arrière lorsque le processus a trop modifié l'image originale ou d'appliquer un autre traitement à partir d'une image intermédiaire afin de vérifier si les résultats obtenus peuvent être améliorés.

En résumé, les besoins des utilisateurs de cette application d'imagerie médicale sont les suivants :

- Faire le lien entre les images résultant de traitements et l'image originale, et connaître pour chaque image par quelles opérations elle a été obtenue.
- Arrêter une série d'opérations et pouvoir la reprendre là où la série avait été stoppée.

1. Référence du projet au Venezuela : s1-9500710.

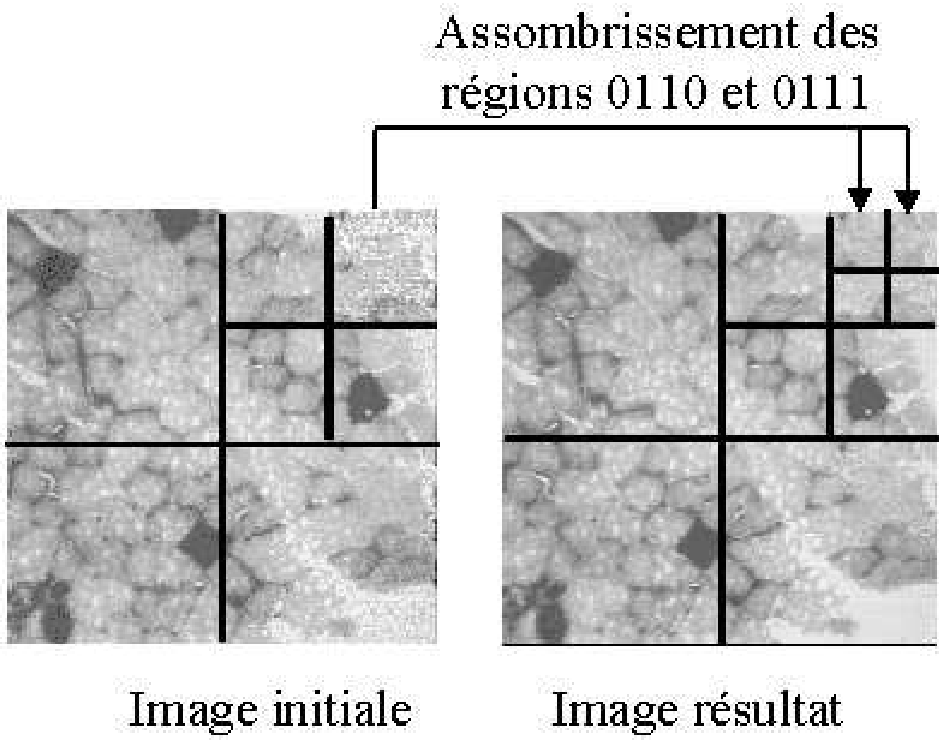


FIG. 3.1 – **Un exemple de modification de régions d'image.**

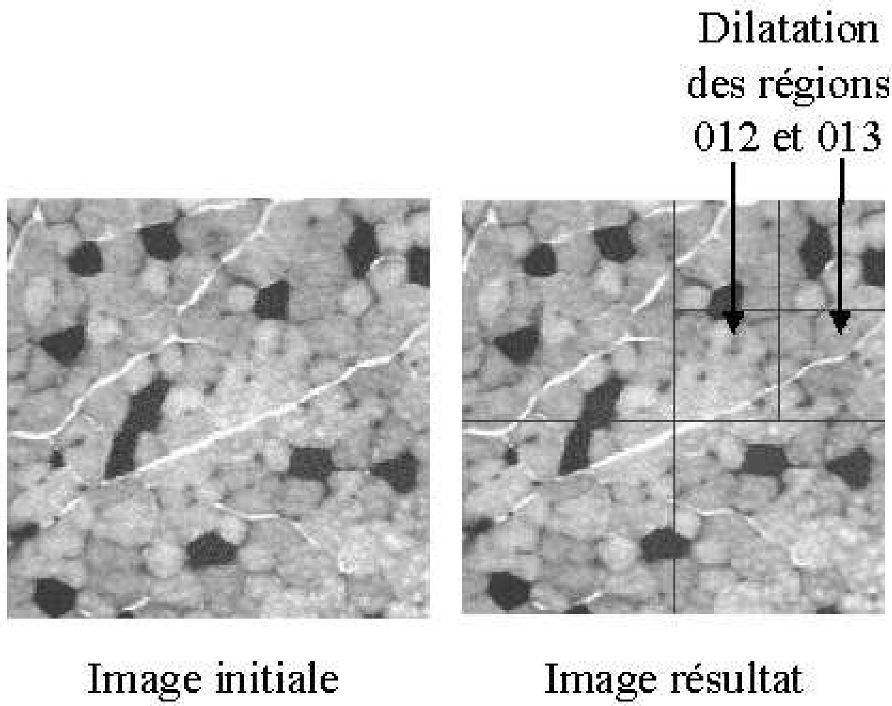


Fig. 3.2 – **Un exemple de dilatation de régions d'image.**

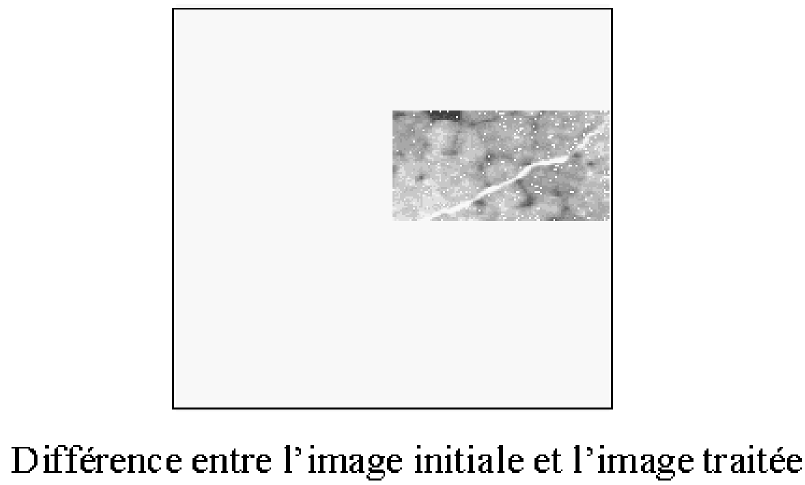


Fig. 3.3 – **La différence entre les images de la figure 3.2.**

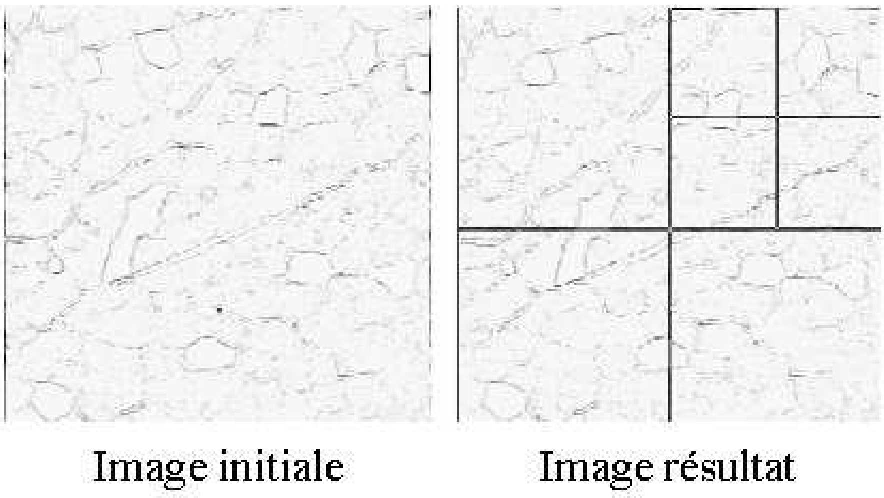


FIG. 3.4 – **La détection des bords sur les complémentaires des images de la figure 3.2.**

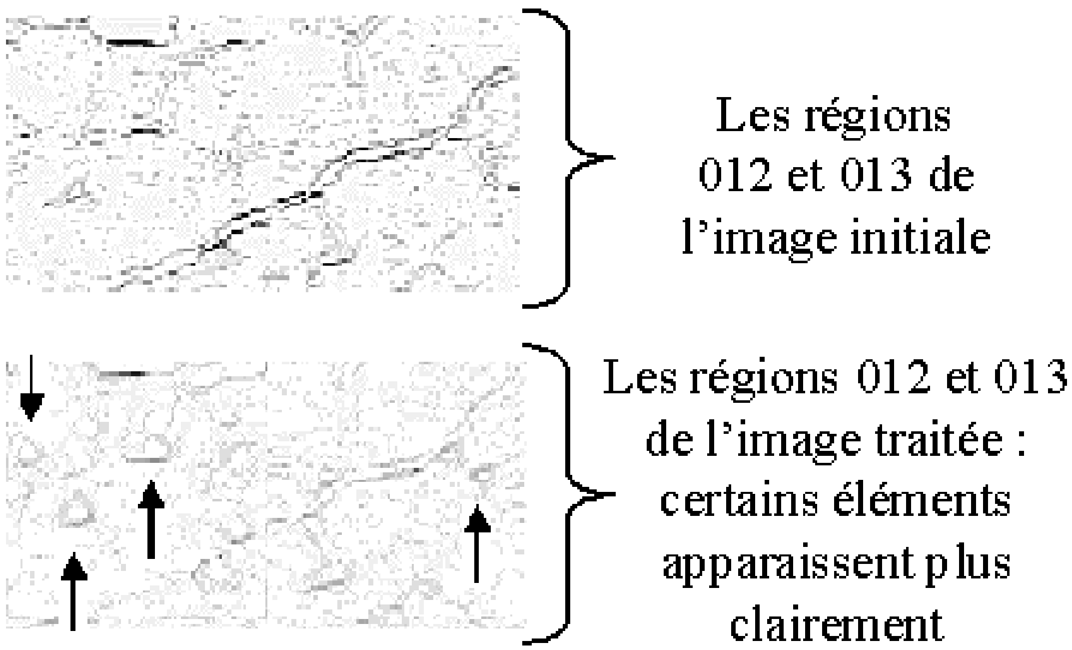


FIG. 3.5 – **La détection d'objets après la dilatation des images de la figure 3.4.**

- Stocker temporairement les différents états obtenus, pour pouvoir en sélectionner "les meilleurs", du point de vue de l'utilisateur, ou permettre des calculs intermédiaires.
- Conserver une trace de toutes les opérations effectuées pour obtenir une image, mais avoir la possibilité de supprimer les images intermédiaires. L'utilisateur peut souhaiter par exemple ne conserver que l'image originale, l'image finale, et éventuellement quelques images intermédiaires, mais pouvoir retrouver la liste de toutes les opérations effectuées.
- Faire des retours en arrière dans une série d'opérations de traitement sur une image.
- Modifier un petit morceau d'image et le replacer dans l'image entière.
- Créer une image à partir de morceaux de l'image traités différemment.
- Suivre l'évolution d'une région de l'image en fonction des traitements appliqués.
- Grouper les images représentant une même réalité pour leur appliquer des traitements similaires.

Conclusion

Les besoins présentés dans ce chapitre ne sont pas restreints au domaine de l'imagerie médicale. Les mécanismes de "mise à jour" (ou traitement) d'images peuvent s'appliquer par exemple à la création d'images de synthèses [LLM98]. Le suivi de l'évolution d'une région dans différentes images est une opération très recherchée dans le domaine spatio-temporel, où il est par exemple intéressant d'observer le mitage (construction d'habitations) de certaines régions territoriales sur des images satellites.

Les images manipulées par les biologistes de l'Université Centrale du Venezuela sont organisées à l'aide d'une structure particulière, l'arbre quaternaire, qui permet de découper récursivement une image en carrés réguliers ou régions. Cette structure est présentée dans le chapitre suivant.

Chapitre 4

Représentation d'images similaires par des arbres quaternaires

Résumé

Ce chapitre décrit rapidement les différentes caractéristiques visuelles des images (couleur, texture, contour) et présente un état de l'art sur les arbres quaternaires. Un arbre quaternaire est une structure de données permettant de représenter des images à deux dimensions. Les images sont découpées en quadrants ou carrés réguliers selon un critère de découpage donné (l'homogénéité de la couleur des pixels, la texture, etc.). Un arbre quaternaire peut être implanté de manière hiérarchique, à l'aide de pointeurs, ou de manière linéaire, sous forme d'une liste de valeurs. Certaines opérations de base de traitement d'image (ex. le complémentaire d'une image, l'union ou l'intersection de deux images) sont applicables sur des images organisées en arbre quaternaire. Plusieurs approches proposent d'optimiser le stockage d'images, en appliquant des mécanismes de chevauchement sur les arbres quaternaires représentant les images. Ces mécanismes peuvent s'appliquer quelle que soit la manière avec laquelle les arbres quaternaires ont été implantés. Ces approches sont principalement orientées vers la gestion et le compactage d'images de séquences vidéo.

Introduction

La représentation des images joue un rôle important dans les applications de traitement d'images. L'arbre quaternaire [FB74], ou *quad-tree*, est une structure de données efficace permettant de représenter les images à deux dimensions. Il s'agit d'une structure, basée sur une décomposition récursive de l'espace [DRS80, Sam84, Sam89]. Un arbre quaternaire découpe l'image en carrés réguliers, selon un critère particulier, comme par exemple l'homogénéité de la couleur des pixels. En plus de la représentation des images, le traitement d'images nécessite l'application de diverses combinaisons d'opérations. Les opérations sur les pixels, par exemple la modification de la couleur d'un ou plusieurs pixels, ainsi que l'intersection et l'union de plusieurs images sont les opérations de base, à partir desquelles sont construites la majorité des opérations utilisées en traitement d'images. Elles sont applicables directement sur des images organisées en arbre quaternaire [SC94], lorsque ceux-ci sont implantés sous forme de structure hiérarchique.

L'organisation de ce chapitre est la suivante. La section 4.1 présente quelques généralités sur les images. La représentation des images par des arbres quaternaires et les opérations applicables à ce type d'images sont décrites en 4.2. Puis, plusieurs approches de représentation de séquences d'images à l'aide d'arbres quaternaires sont présentées en 4.3.

4.1 Généralités sur les images

Il existe plusieurs mécanismes pour décrire une image [Gob97] :

- **La description textuelle** : il s'agit de mots-clé décrivant l'image en entier ou les objets qui la composent.
- **La description sémantique** : elle correspond à une représentation sémantique de haut niveau (modèle objet par exemple) décrivant le contenu de l'image.
- **La description basée sur les caractéristiques visuelles** : il s'agit d'informations sur les caractéristiques comme la couleur, la forme, ou la texture, décrivant l'image ou les objets qui la composent.

Les caractéristiques visuelles des images peuvent être classées selon plusieurs niveaux d'abstraction. Le premier niveau concerne les pixels de l'image (ou *picture element*), c'est-à-dire les points élémentaires figurant sur l'écran. Un pixel est la plus petite particule visible par l'intermédiaire de signaux numériques. Les images monochromes contiennent des pixels dont la valeur est 0 ou 1, indiquant si le pixel est noir ou blanc. Les images à haute résolution contiennent des pixels de 8, 16, 24 ou 32 octets [KB96, MM99]. Le deuxième niveau d'abstraction concerne les caractéristiques de l'image en général comme la couleur, la texture ou la forme de ses différentes régions. Finalement, le dernier niveau concerne les objets sémantiques contenus par l'image, ainsi que leurs attributs et leurs relations avec les autres objets de l'image.

Cette section décrit les différentes caractéristiques des images. La sous-section 4.1.1 présente rapidement les différents formats d'image. La sous-section 4.1.2 décrit les principales caractéristiques visuelles des images.

4.1.1 Format

Il existe plusieurs formats d'images [BYRN99]. Les formats les plus simples consistent en une représentation des images basée sur les pixels (*bit-mapped*), comme les formats XBM, BMP ou PCX. Cependant, ces formats sont très coûteux en espace. Par exemple, l'image représentant la copie d'un écran de 256 couleurs peut occuper plus d'un Mega-octets [BYRN99].

Du fait de la redondance d'information très présente dans les images, plusieurs formats de compactage ou de compression ont été proposés. Le plus connu est le format GIF (*Compuserve's Graphic Interchange Format*) [Com87]. Ce format est principalement approprié aux images en noir et blanc, en niveau de gris ou contenant peu de couleurs (moins de 256). Pour augmenter le taux de compression, des formats de compression avec perte ont également été proposés, comme JPEG (*Joint Photographic Experts Group*) [Wal89, Wal91]. JPEG travaille sur des images de 24 bits soit 16 millions de couleurs simultanées ou 8 bits pour les images en niveaux de gris. Ce format élimine notamment les parties de l'image qui ont le moins d'impact à l'œil nu. Les formats tels que GIF ou JPEG sont typiquement destinés à la visualisation ou au transfert de données via le réseau, c'est-à-dire pour des applications dont l'objectif principal est l'affichage [DL93].

Il existe d'autres formats comme le format TIFF (*Target Image File Format*) [AM88] utilisé pour les échanges de document entre différentes applications et différentes plates-formes [BYRN99]. Le format Sun raster (RAS) a été proposé par Sun pour manipuler des images sous Unix. Son utilisation est comparable à celle qui peut être faite du format TIFF, à savoir un stockage sans perte des images ou un format de travail pour des manipulations d'image [DL93]. Cependant, le taux de compression du format RAS est moins important que celui de TIFF.

Certains formats ont été proposés pour répondre aux besoins d'applications particulières, comme par exemple le format PNG (*Portable Network Graphics*) pour les images transmises sur Internet.

4.1.2 Caractéristiques visuelles

Les principales caractéristiques visuelles des images sont la couleur, la texture et la forme (*shape*). Cette sous-section présente rapidement chacune de ces caractéristiques.

a) Couleur

L'attribut le plus significatif d'une image est la couleur [Nas97]. Il existe plusieurs techniques permettant de déterminer et représenter la couleur d'une image.

Par exemple, il est possible d'utiliser trois composantes : le rouge, le vert et le bleu (ou RGB pour *red*, *green*, *blue*). Chaque pixel d'une image couleur est représenté par une combinaison linéaire de ces trois composants [Fal96]. Un pixel est représenté par un triplet (r, g, b), où r, g et b correspondent respectivement à l'intensité de rouge, de vert et de bleu, avec $r, g, b \in [0, 255]$. La couleur blanche est représentée par le triplet (255,255,255), la couleur noire par le triplet (0,0,0), le triplet (200,60,60) correspondant à la couleur rose.

Une représentation équivalente est l'utilisation des composantes telles que la nuance, la saturation et l'intensité (HSI : *Hue-Saturation-Intensity*). La nuance correspond aux couleurs considérées comme pure [MM99] (la pureté des couleurs dépendant de l'espace de couleur choisi), alors que la saturation correspond à la quantité de blanc ajoutée à la couleur pure pour obtenir la couleur du pixel. Pour plus de détails, le lecteur est invité à lire [MG98, YM98a].

La couleur des images peut également être caractérisée par un histogramme de couleurs, dont l'axe horizontal représente les différentes couleurs et l'axe vertical représente, pour chaque couleur, les pourcentages de pixels ayant cette couleur [YM98a]. Cet histogramme correspond à la distribution des couleurs dans l'image.

Il est possible d'utiliser un vecteur à k dimensions pour représenter une image, le $i^{\text{ème}}$ composant correspondant au pourcentage de pixels de la $i^{\text{ème}}$ couleur. Le système QBIC (*Query By Image Content*) [FSN+95], par exemple, utilise des vecteurs de 64 à 256 dimensions, chaque dimension représentant un intervalle de couleurs. La similarité des histogrammes de couleurs est calculée en fonction de ces vecteurs. Pour plus de détails, le lecteur est invité à consulter [Fal99]. D'autres mesures de similarités des images par les couleurs sont décrites dans [ABH97]. Plusieurs techniques d'indexation de la couleur des images ont été proposées dans [LOT94, MG98, OTCH98].

b) Texture

La texture d'une image correspond à un motif visuel contenant un grand nombre d'éléments visibles denses et organisés [AY99]. Les éléments visibles sont un ensemble de pixels ayant tous un certain niveau de gris ou des caractéristiques identiques de couleur. Les propriétés associées à la texture sont les contrastes, la grosseur et la direction. La texture est très utilisée pour caractériser les images géographiques [SZB99].

La texture, à la différence de la couleur, n'est pas un attribut de pixel mais une caractéristique de région d'image [AY99]. La segmentation permet de trouver les régions d'une image dont la texture est homogène.

Il existe plusieurs techniques permettant d'indexer et de capturer la similarité des textures d'image : les fractals, les transformées de Fourier, les ondelettes ou la morphologie mathématique. Pour plus de précisions, le lecteur est invité à lire [Mit97, AY99, SZB99].

Plusieurs approches permettent de faire de la recherche d'image par la texture [SC94, FSN+95, SZB99]. Des exemples de résultats de recherche par similarité de texture sont présentés dans [Nas97].

c) Forme

Le terme de "forme", également appelé contours (ou *shape*), est plus souvent utilisé pour caractériser les objets contenus dans une image plutôt que l'image elle-même [Mit97]. Il s'agit d'une caractéristique significative dans de nombreuses applications, comme l'imagerie médicale [Nas97].

La forme d'un objet dans une image est décrite par plusieurs caractéristiques telles que la région, le périmètre, ou les axes d'orientation [YM98a]. Comme la couleur, la forme peut être représentée par un vecteur à plusieurs dimensions [YM98a]. Il faut en général plus d'une centaine de paramètres pour représenter explicitement cette caractéristique [Fal96, MAK97]. Le système QBIC [FSN+95] utilise, par exemple, 20 dimensions pour représenter la forme d'un objet d'une image. Pour plus de détails sur le méthodes mathématiques permettant de déterminer la forme d'un objet dans une image, le lecteur est invité à consulter [DSBM97].

La recherche d'image par la forme fait partie des problèmes les plus complexes de recherche d'images [KSF+96, AY99]. La recherche de contours ou de forme d'objets dans une image est en effet plus complexe que la quantification de la couleur ou de la texture [MAK97]. L'indexation de cette caractéristique et les mesures de similarité sont également complexes à mettre en place car la forme est une notion difficilement indépendante de la perception humaine [DBP97].

d) Représentation des caractéristiques visuelles d'une image

Pour capturer le contenu d'une image, on utilise ce que [Mit97, Nas97] appellent des *signatures d'images*. Une signature est une manière de représenter de façon numérique une caractéristique d'image. Les signatures dites de bas-niveau sont liées à la couleur, à la texture ou à la forme. Il s'agit d'histogramme de couleur, de transformées en ondelettes ou de transformées de Fourier par exemple [NBMM98]. Les signatures de plus haut niveau sont calculées à partir d'une modélisation complexe du contenu de l'image ou d'une analyse statistique d'une base d'images [NBMM98].

Chaque caractéristique d'une image pouvant être quantifiée, il est possible de représenter une image par un vecteur à plusieurs dimensions [YM98a]. Chaque signature d'image représente un point de ce vecteur [Mit97]. Les caractéristiques d'image sont généralement combinées pour représenter au mieux l'image [NBMM98].

Pour extraire plus facilement le contenu des images, certaines approches [LOT94, SC94, MBNW99] utilisent des arbres quaternaires. La structure de l'arbre quaternaire est décrite en détail dans la section suivante.

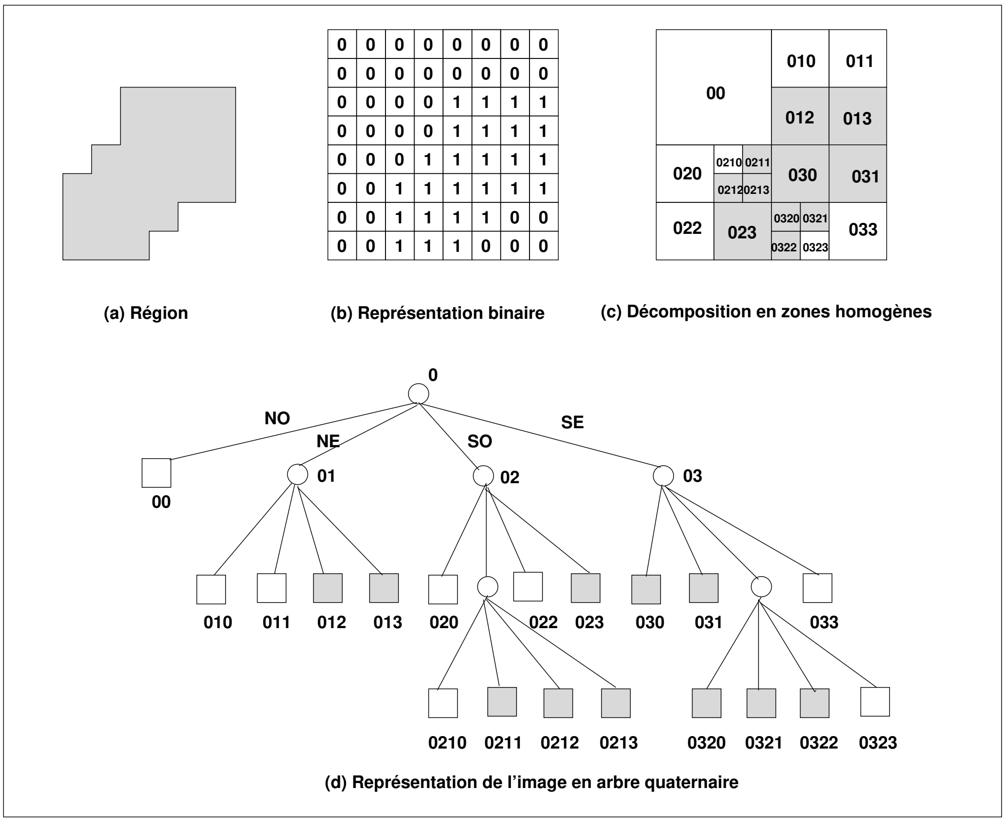
4.2 Organisation d'images en arbre quaternaire

Cette section présente, en 4.2.1, les différentes manières d'implanter un arbre quaternaire : sous forme hiérarchique ou linéaire. Les différents critères de découpage d'une image en arbre quaternaire sont présentés en 4.2.2. Les opérations de base de traitement d'images réalisables sur les images organisées en arbre quaternaire sont décrites en 4.2.3.

4.2.1 Définition générale et implantation d'un arbre quaternaire

a) Implantation hiérarchique

Il existe des arbres quaternaires sur divers types de données, comme les courbes, les points, les surfaces, ou les volumes [Sam89]; une description détaillée de cette structure est présentée dans [Sam84]. Le plus connu est l'arbre quaternaire sur les régions, qui permet de découper une image en zones (ou régions) répondant aux mêmes caractéristiques : homogénéité de la couleur des pixels ou homogénéité de la texture, par exemple. Plusieurs critères de découpage sont envisageables



(a) Région
(b) Représentation binaire
(c) Décomposition en zones homogènes
(d) Représentation de l'image en arbre quaternaire

FIG. 4.1 – **Une image de taille $2^3 * 2^3$ organisée en arbre quaternaire [Sam84].**

(voir section 4.2.2).

L'image est découpée récursivement en quatre quadrants ou carrés réguliers tels que chaque quadrant soit représenté par un nœud de l'arbre quaternaire. La racine représente l'image dans sa totalité. Elle a quatre nœuds fils représentant les quatre principaux quadrants, N-O (Nord-Ouest), N-E (Nord-Est), S-O (Sud-Ouest) et S-E (Sud-Est). Chaque nœuds fils possède à son tour quatre fils, représentant ainsi les seize sous-quadrants de l'image initiale. Puis, ces quadrants sont à leur tour découpés en quatre, formant 64 nœuds et ainsi de suite. Les nœuds feuilles de l'arbre quaternaire ne sont pas tous au même niveau. Ils représentent les régions de l'image répondant à un même critère de découpage, qui peut être la couleur de la région par exemple. La figure 4.1 donne un exemple d'image binaire (contenant des pixels noirs et blancs) organisée en arbre quaternaire dont le critère de découpage est la couleur.

Il existe plusieurs fonctions permettant de savoir à quelle partie de l'image correspond un nœud de l'arbre quaternaire, et inversement. Ces fonctions utilisent une identification particulière des nœuds. La racine de l'arbre quaternaire, représentant l'image tout entière, est identifiée, par exemple par 0. Ses fils, représentant les quatre premiers quadrants de l'image, ont pour identificateurs respectifs $00, 01, 02$ et 03. Chaque fils du nœud 00 a pour identificateur $00x$, $x \in [0, 3]$, chaque fils du nœud 01 a pour identificateur $01x$, $x \in [0, 3]$, etc. Cette numérotation est employée dans la figure 4.1. Il s'agit d'une numérotation en Z. D'autres types de numérotation sont présentés dans [Sam84].

Un arbre quaternaire hiérarchique est une structure qui permet de faire rapidement des opérations sur les images. Il permet, par exemple, de faire des zooms sur une région d'image en stockant, dans les nœuds internes, le niveau de gris représentant la moyenne des couleurs des nœuds fils. Une lecture horizontale, c'est-à-dire niveau par niveau, de l'arbre quaternaire permet d'avoir une idée générale de l'image. Puis, plus on descend dans les niveaux de l'arbre plus l'information est précise. De même, il est possible de traiter les sous-arbres d'un même arbre quaternaire à l'aide d'algorithmes parallèles. Les arbres quaternaires permettent également d'accélérer les opérations sur plusieurs images, comme l'union, l'intersection ou la comparaison (voir section 4.2.3) en traitant en parallèle différents sous-arbres. Ces opérations s'effectuent par comparaison des sous-arbres de chaque arbre quaternaire.

Toutefois, l'implantation d'un arbre quaternaire sous forme de structure hiérarchique est coûteuse en mémoire. En effet, [Ste86] montre que 90% de la mémoire utilisée pour stocker un arbre quaternaire hiérarchique est occupée par les pointeurs, lorsque deux octets sont utilisés pour stocker chaque pointeur et un octet pour stocker chaque identificateur de nœud. Ainsi, pour palier à l'occupation abusive de la mémoire par les pointeurs, des chercheurs [KE80, Lin97, TK91] ont proposé des méthodes de stockage linéaire d'arbre quaternaire.

b) Implantation linéaire

La représentation des arbres quaternaires sous forme linéaire a été proposée dans l'optique de supprimer la place occupée pour le stockage des pointeurs. Une représentation linéaire de cette structure consiste en une liste de valeurs qui conserve implicitement la structure hiérarchique de l'arbre. Seules les valeurs des nœuds feuilles sont stockées dans la liste. Il existe différentes manières de stocker ces valeurs [Sam84]. Deux approches sont présentées dans ce document, celle de Kawaguchi et Endo [KE80], basée sur une lecture de l'arbre quaternaire en profondeur d'abord et celle de Lin [Lin97], basée sur une lecture en largeur d'abord.

La méthode de stockage linéaire, proposée par [KE80], code chaque nœud feuille noir par un N, et chaque nœud feuille blanc, par un B. La hiérarchie de la structure est conservée par l'utilisation de parenthèses; une parenthèse code un nœud interne.

Exemple 4.1 L'arbre quaternaire de l'image de la figure 4.1 (voir (d) sur la figure) est codé, via cette méthode, par la séquence : $(B(BBNN(B(BNNNBN(NN(NNNBB$. La première parenthèse indique que la racine de l'arbre quaternaire est un nœud interne ; l'image a été découpée en quatre quadrants principaux identifiés de 00 à 03 (voir (c) sur la figure). Le premier B code la couleur du nœud 00 de l'arbre quaternaire, qui est blanc. La deuxième parenthèse signifie que le nœud 01 a été découpé en quatre. La séquence $BBNN$ code les couleurs des nœuds fils de 01. La troisième parenthèse indique que le nœud 02 est interne, et le B qui suit signifie que le nœud 020 est une feuille blanche. La séquence $(BNNNBN$, pour être comprise, doit être décomposée en trois parties : (1) la parenthèse signifie que le nœud 021 est interne, (2) la sous-séquence $BNNN$ code les couleurs des nœuds fils de 021, (3) les deux lettres restantes, BN, correspondent aux couleurs respectives des nœuds 022 et 023.

Le code *CBLQ* (*Constant Bit-lenght Linear Quadtree*), proposé par [Lin97] en 1997, utilise un autre type de stockage linéaire de l'arbre quaternaire, en parcourant l'arbre en largeur d'abord. Chaque nœud feuille noir est codé par un 1. Inversement, chaque nœud feuille blanc est codé par un 0. Puis, chaque nœud interne, sauf la racine, est codé par 2, si au moins un de ses fils est non feuille, ou par 3, si tous ses fils sont feuilles. La racine n'est pas codée.

Exemple 4.2 En utilisant ce code, l'arbre quaternaire de la figure 4.1 est représenté par la séquence : *0322 0011 0301 1130 0111 1110*. Les quatre premiers chiffres représentent le codage des nœuds du premier niveau de l'arbre quaternaire (identifiés par 00, 01, 02 et 03). Le nœud identifié par 00 est blanc, il est donc codé par un 0. Le nœud identifié par 01 est interne et tous ses fils sont feuilles, il est donc codé par un 3. Les nœuds identifié par 02 et par 03 sont internes et possèdent des fils qui sont eux-mêmes nœuds internes, ils sont donc codés par un 2. La séquence *0011* correspond au codage des nœuds 010 à 013. La séquence *0301* code les nœuds 020 à 023, etc.

L'implantation linéaire des arbres quaternaires a été proposée pour permettre un stockage efficace des images organisées en arbre quaternaire, sans occuper la mémoire par les pointeurs du stockage hiérarchique. Ce type de stockage permet également de transmettre plus rapidement et à moindre coût des images sur un réseau. L'inconvénient ce type d'implantation est qu'il faut reconstruire l'image pour la visualiser ou la modifier, le temps de reconstruction dépendant fortement de la complexité du code utilisé. Les opérations sur les images organisées en arbre quaternaire, décrites dans la section 4.2.3, ne sont pas directement applicables sur les images dont l'arbre quaternaire est implanté linéairement. Il faut reconstruire l'arbre quaternaire hiérarchique associé pour pouvoir effectuer l'opération.

4.2.2 Critères de découpage d'une image en quadrants

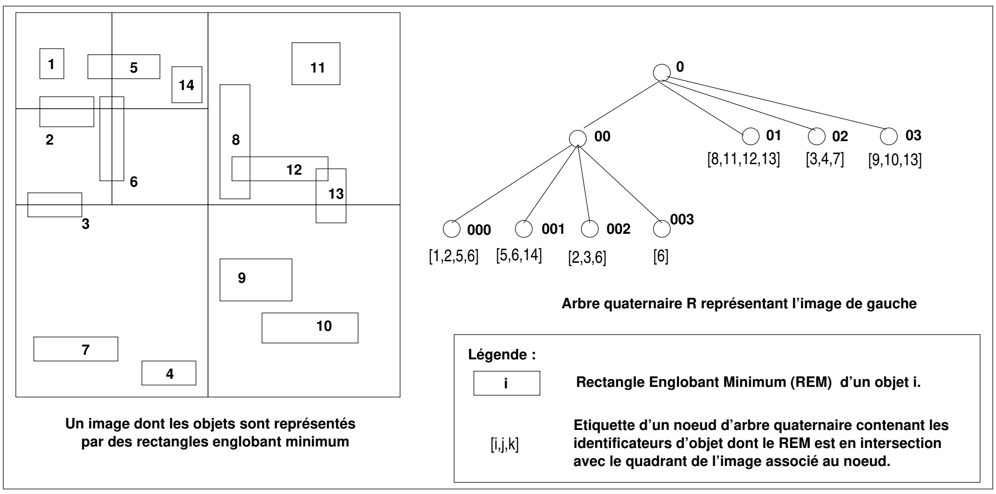


FIG. 4.2 – **Un exemple d'arbre quaternaire R de seuil égal à 4.**

Dans l'exemple de la figure 4.1, le critère de découpage de l'image, en vue de son organisation en arbre quaternaire, est l'homogénéité de la couleur des quadrants. Lorsque l'image est en niveaux de gris, et non plus en noir et blanc comme dans l'exemple, le découpage de l'image par ce critère peut aller jusqu'au niveau des pixels. Il est possible de fixer un seuil d'homogénéité pour limiter le découpage. Cependant, l'homogénéité des couleurs des quadrants n'est pas un critère de découpage conseillé pour les images autres que les images en noir et blanc. Lorsque l'image est en niveaux de gris, le critère de découpage peut être par exemple l'homogénéité de la texture (voir section 4.1.2), comme dans l'approche de [SC94].

Lorsque les objets composant une image peuvent être déterminés et représentés par un rectangle englobant minimum (*Minimum Bounding Rectangle* ou MBR), il est possible de découper l'image en quadrants tels que le nombre maximum de rectangles englobants, pouvant être en intersection avec la région correspondante de l'image, est fixé. L'arbre quaternaire porte dans ce cas le nom d'arbre quaternaire R (pour rectangle) [PRS95, SVP$^+$96, Gün98]. La figure 4.2 présente un exemple d'arbre quaternaire R, dont le nombre maximum (ou seuil) de rectangles englobants minimums pouvant être en intersection avec les quadrants de l'image est fixé à 4.

Dans l'application d'imagerie médicale, présentée dans le chapitre 3, l'utilisateur choisit la profondeur de découpage de ses images, en fonction des régions sur lesquelles il souhaite travailler ou appliquer des traitements. Dans la figure 3.1, l'utilisateur a découpé l'image en quatre quadrants, puis la région 01 en quatre quadrants, et finalement la région 011 en quatre, afin de pouvoir assombrir les régions 0110 et 0111. La représentation en arbre quaternaire de cette image est illustrée sur la figure 4.3.

De même qu'il est possible de découper une image en fonction de différents critères, il est possible d'associer plusieurs types d'information aux nœuds de l'arbre quaternaire représentant l'image. Lorsque le critère de découpage est la couleur, les nœuds feuilles de l'arbre peuvent contenir uniquement le code de la couleur, par exemple l'intensité de rouge, de vert et de bleu (voir section 4.1.2). Les auteurs de [LOT94] stockent les histogrammes de couleur des quadrants de l'image dans les nœuds de l'arbre quaternaire. Dans le cas de l'arbre quaternaire R, l'information contenue dans les nœuds feuilles est un ensemble d'identificateurs d'objet (voir figure 4.2). Les auteurs de [SC96] font pointer les nœuds feuilles des arbres quaternaires vers une représentation de la texture en ondelettes[1].

4.2.3 Opérations applicables aux images organisées en arbre quaternaire

Il existe plusieurs opérations de traitement d'images réalisables sur des images organisées en arbre quaternaire. Seuls trois types sont étudiés dans cette section : les opérations sur les pixels d'une image, l'intersection et l'union de deux images. D'autres opérations, basées sur les opérateurs morphologiques, comme la dilatation ou l'érosion de régions d'images, sont présentées dans [MC99]. Les opérations présentées dans cette section ne s'appliquent qu'aux images organisées en arbre quaternaire dont les nœuds feuilles contiennent un morceau d'image (voir figure 4.3) ou un quadrant homogène (voir figure 4.1). Elles ne s'appliquent pas aux images organisées en arbre quaternaire R (voir figure 4.2).

1. Les ondelettes (ou *wavelet*) correspondent à une méthode mathématique utilisée en traitement du signal et permettant de compresser les images. Pour plus de détails, le lecteur est invité à consulter [MPA96, Dav98, Mal99].

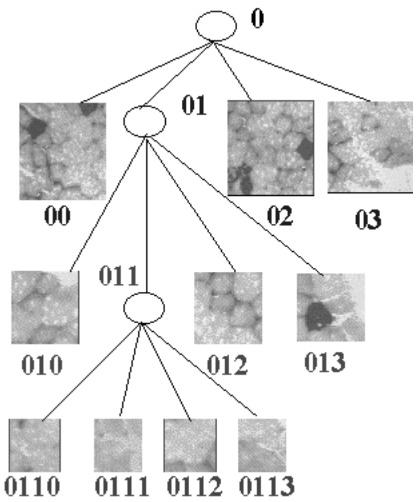


FIG. 4.3 – **L'image de la figure 3.1 organisée en arbre quaternaire.**

a) Opérations sur les pixels d'une image

Les opérations sur les pixels d'une image correspondent à des modifications de l'image pixel par pixel. Lorsque l'image est organisée en arbre quaternaire, tout l'arbre est parcouru, et chaque nœud feuille est mis à jour.

L'opération *Complémentaire*, par exemple, est une opération sur les pixels d'une image, qui consiste à inverser la couleur de chaque pixel de l'image. Chaque pixel a une valeur comprise entre 0 et 255, 255 correspondant à blanc, 0 à noir et les valeurs intermédiaires aux différents niveaux de gris. La valeur complémentaire d'un pixel est égale au résultat de l'opération *255 - (valeur du pixel)*. La figure 4.4 donne un exemple d'application de l'opération Complémentaire à une image binaire (en noir et blanc) organisée en arbre quaternaire.

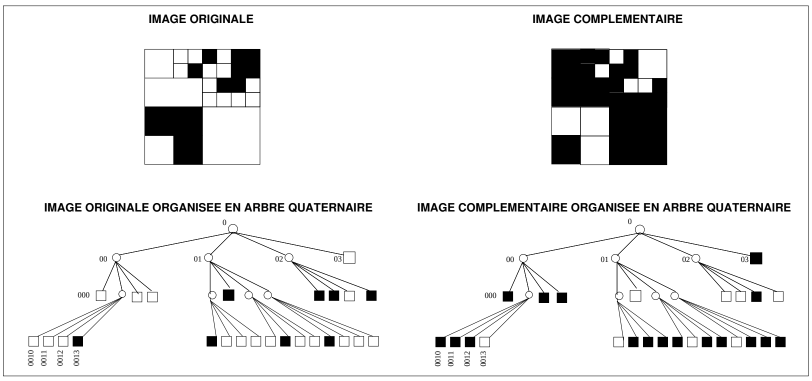


FIG. 4.4 – **Le complémentaire d'une image organisée en arbre quaternaire.**

b) Intersection et union de deux images

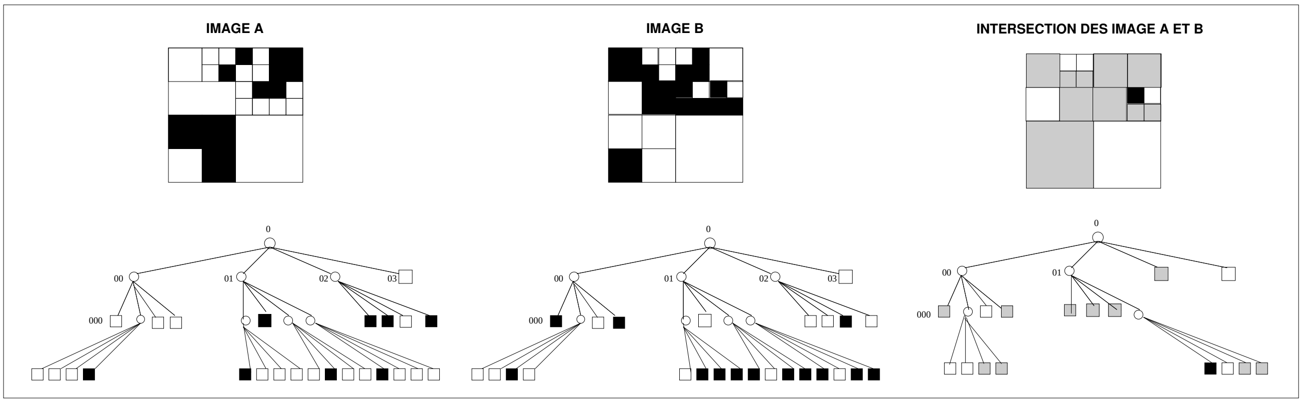


FIG. 4.5 – **L'intersection de deux images organisées en arbre quaternaire.**

Pour faciliter la compréhension du lecteur, les opérations d'intersection et d'union sont expliquées dans un cas simple d'images en noir et blanc. Pour plus de détails, le lecteur est invité à consulter [Sam84, Sam89].

L'intersection de deux images, lorsqu'elle sont organisées en arbre quaternaire, consiste à comparer les nœuds des différents arbres quaternaires. De l'intersection de deux images résulte une nouvelle image et donc un nouvel arbre quaternaire, lorsque les images sont organisées par cette structure.

De l'intersection de deux images A et B résulte une image C dont les nœuds de l'arbre quaternaire sont tels que :

- Si un nœud est noir (resp. blanc) dans les arbres quaternaires des images A et B, il est noir (resp. blanc) dans l'arbre quaternaire de l'image C.
- Si un nœud n'est pas de la même couleur dans les arbres quaternaires de A et B, il est indéfini, c'est-à-dire de la couleur du fond (par ex. gris), dans l'arbre quaternaire de C.
- Lorsqu'un nœud est feuille dans un des arbres quaternaires, par exemple celui de l'image A, et interne dans l'autre, par exemple celui de l'image B, il doit être temporairement divisé dans l'arbre quaternaire de A, pour permettre la comparaison. Le nœud devient interne dans l'arbre quaternaire de C.

La figure 4.5 donne un exemple d'intersection de deux images, ainsi que leur représentation respective par des arbres quaternaires. Dans cet exemple, la couleur du fond est grise.

L'opération d'union de deux images reprend les mêmes principes à l'exception que lorsqu'un nœud est noir dans un des deux arbres quaternaires il devient noir dans l'arbre quaternaire de l'image résultat. Lorsque tous les nœuds fils d'un nœud interne n sont de la même couleur (noire), le nœud n devient noir et ses nœuds fils disparaissent. La figure 4.6 présente un exemple d'application de cette opération. Un exemple d'union de deux images en niveaux de gris est présenté dans la figure 5.22 (voir page 80).

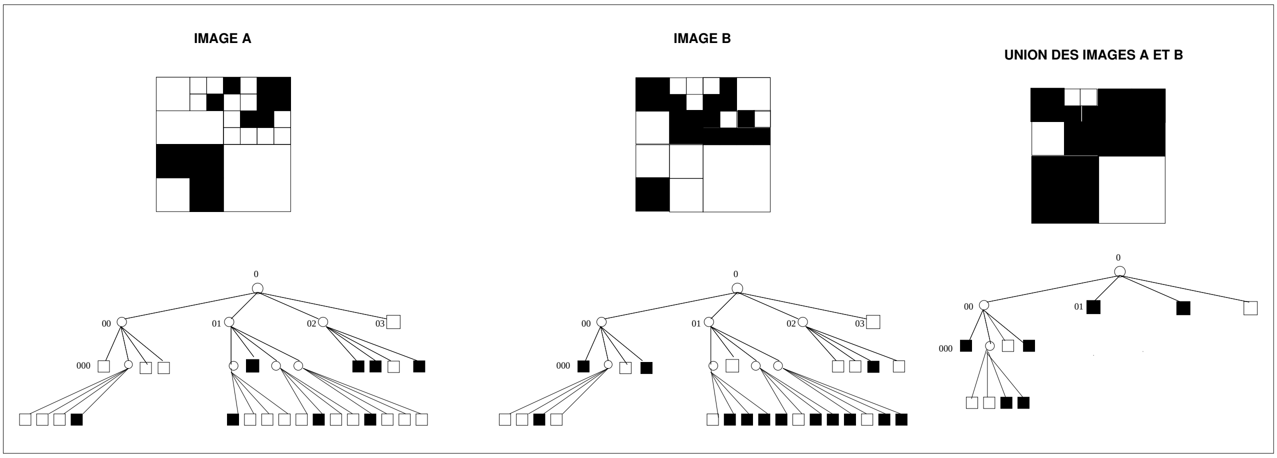


FIG. 4.6 – **L'union de deux images organisées en arbre quaternaire.**

4.3 Chevauchement d'arbres quaternaires

Plusieurs approches [VME93, Lin97, TVM98] ont proposé des mécanismes de représentation de séquences d'images similaires organisées en arbres quaternaires, notamment dans le cas de la gestion de séquences vidéo d'images binaires où deux images successives ont très peu de différences entre elles. Ces approches appliquent les mécanismes de chevauchement (ou *Overlapping*), décrits dans la partie A, sur les arbres quaternaires, afin de partager le plus grand nombre de sous-arbres communs entre arbres quaternaires. Il s'agit donc de faire partager le maximum de pixels ou de régions entre images. Ce partage peut exister quelle que soit la manière, hiérarchique ou linéaire, d'implanter les arbres quaternaires.

Cette section décrit ces différentes approches et les compare au travers d'un même exemple simple (voir figures 4.7) de quatre images binaires. Les images de cet exemple ont été choisies de sorte qu'elles correspondent à des images pouvant être générées par les utilisateurs de l'application d'imagerie médicale décrite dans le chapitre 3. Chaque image correspond au résultat d'un traitement local sur une région d'image. L'application des mécanismes de chevauchement sur cet exemple permet de montrer que les approches présentées dans cette section, ne répondent pas complètement aux besoins des utilisateurs d'applications de traitement d'images.

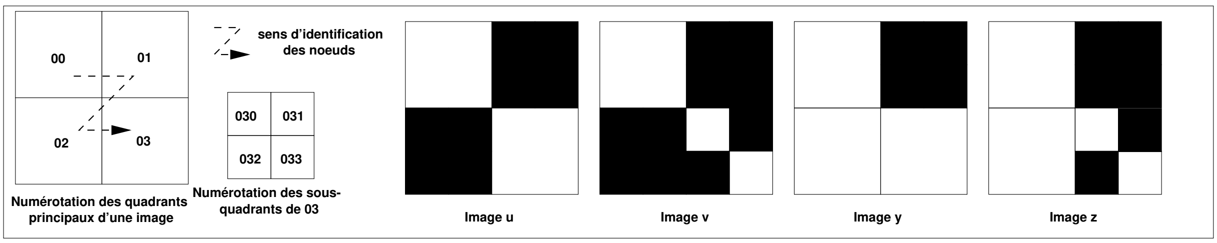


FIG. 4.7 – **Un exemple d'images.**

4.3.1 Chevauchement d'arbres quaternaires implantés de manière hiérarchique

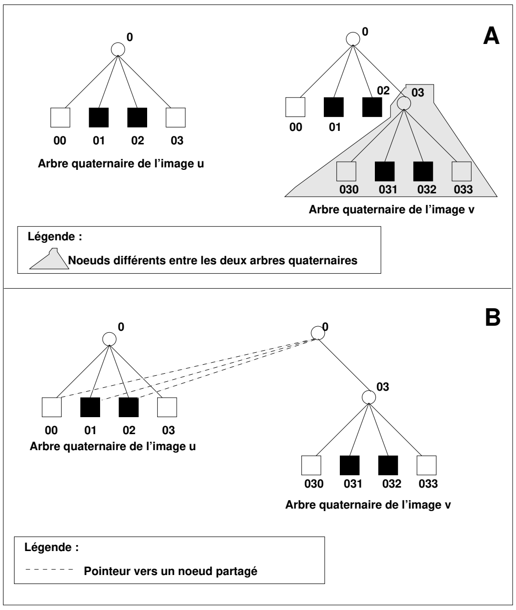


FIG. 4.8 – **Le chevauchement des arbres quaternaires de deux images.**

Dans l'approche de [VME93], chaque image est organisée en arbre quaternaire implanté de manière hiérarchique. Les images, appartenant en général à une vidéo, sont organisées en séquences linéaires. Les mécanismes de chevauchement classiques [BHK85], expliqués dans la partie A, sont appliqués aux arbres quaternaires. Un arbre quaternaire partage plusieurs nœuds avec l'arbre quaternaire correspondant à l'image précédente dans la séquence. Ce partage se fait en utilisant des pointeurs vers les nœuds partagés. On dit que l'arbre quaternaire de l'image i chevauche celui de l'image précédente, $i-1$. Lorsqu'un nœud de même identificateur a une valeur différente dans deux arbres quaternaires, il y a copie des tous les nœuds du chemin allant du nœud modifié à la racine, dans l'arbre quaternaire de l'image i.

Exemple 4.3 L'application de ces mécanismes sur les deux arbres quaternaires des premières images de l'exemple de la figure 4.7 est illustrée sur la figure 4.8. Les nœuds qui diffèrent entre les arbres quaternaires des images u et v sont grisés (voir partie A en haut de la figure). Le bas de la figure 4.8 représente les arbres quaternaires chevauchées obtenus après partage des parties communes entre les arbres.

Les caractéristiques de l'approche [VME93] sont les suivantes. Le partage des parties communes entre arbres quaternaires permet de minimiser l'espace de stockage des images. Cependant, ce partage ne peut se faire que lorsque des parties communes apparaissent dans deux images successives de la séquence. En particulier, il ne peut pas y avoir de partage de motif qui pourrait apparaître périodiquement parmi les images de la séquence. De plus, la modification d'une image est difficile. Cette opération ne semble d'ailleurs pas avoir été envisagée par les auteurs. En effet, les mécanismes de chevauchement ne permettent pas d'avoir une connaissance a priori sur le partage. Par conséquent, la modification d'une image implique la création d'une nouvelle image et donc d'un nouvel arbre quaternaire, afin de ne pas modifier des parties qui seraient éventuellement partagées par d'autres arbres. La séquence d'images évolue donc par ajout de nouvelles images à la fin de la séquence, c'est-à-dire par la construction de nouveaux arbres quaternaires chevauchés. Toutefois, dans le cas où les éléments sont en mémoire centrale, la lecture d'une image ne prend pas plus de temps lorsque la structure d'arbres quaternaires chevauchés est utilisée que lorsqu'elle ne l'est pas et que les images sont stockées, indépendamment les unes des autres, dans des arbres quaternaires.

L'approche de [VME93] a été étendue au codage linéaire (voir section 4.2.1) dans [TVM98]. Dans cette extension, les identificateurs des nœuds noirs sont stockés dans un arbre B+, et les mécanismes de chevauchement sont appliqués aux arbres B+ représentant le codage linéaire des arbres quaternaires. On retrouve dans les arbres B+ de l'approche [TVM98] les inconvénients, cités dans le paragraphe précédent, concernant le chevauchement des arbres quaternaires.

La sous-section suivante présente des mécanismes de chevauchement d'arbres quaternaires implantés de manière linéaire.

4.3.2 Chevauchement d'arbres quaternaires linéaires

Le code CBLQ de [Lin97], présenté dans la section 4.2.1, a également été étendu afin de représenter des séquences d'images similaires totalement ordonnées dans le but principal de minimiser l'espace, en particulier dans le cas d'images issues de séquences vidéo.

La première et la dernière image de chaque séquence sont codées par le code CBLQ. Puis, toutes les images de la séquence, excepté la première, sont stockées à l'aide du code *CBLQ chevauché*. Ce code correspond aux différences entre deux images successives tel que le code chevauché de l'image I_i corresponde aux différences de I_i avec l'image I_{i-1}. Ce code chevauché se construit comme suit. Les quatre premiers quadrants des deux images sont comparés et sont codés par un A s'ils sont identiques dans les deux images ou un B sinon. Pour chaque quadrant codé par un B, les sous-quadrants sont comparés entre les deux images et sont codés par un A (s'ils sont identiques), un C ou un D (voir table 4.1). Pour plus de détails, le lecteur est invité à consulter [Lin97].

TAB. 4.1 – **Codage des nœuds de quad-tree par le code CBLQ chevauché de [Lin97]**

Quadrant de la première image :	Blanc	Blanc	Découpé	Découpé	Noir	Noir
Quadrant de la deuxième image :	Découpé	Noir	Noir	Blanc	Blanc	Découpé
Code chevauché :	C	D	C	D	C	D

Exemple 4.4 Le codage des quatre images de la figure 4.7 est le suivant :
0110 ⟵ code CBLQ de l'image u.
AAAB ADDA ⟵ code chevauché des images u et v.
AABB C ACCA ⟵ code chevauché des images v et y.
AAAB ADDA ⟵ code chevauché des images y et z.
0103 0110 ⟵ code CBLQ de l'image z.

Pour stocker et restituer une image de la séquence, [Lin97] propose trois opérations :

1. *overlap* : calcule le code chevauché d'une image par rapport à l'image la précédant dans la séquence. $O_i = overlap(I_{i-1}, I_i)$

2. *un_overlap* : calcule, de manière récursive, le code CBLQ d'une image par rapport à l'image précédente. $I_i = un\_overlap(I_{i-1}, O_i)$; cette fonction s'applique récursivement depuis la première image de la séquence.

3. *re_overlap* : calcule, de manière récursive, le code CBLQ d'une image par rapport à l'image suivante. $I_i = re\_overlap(I_{i+1}, O_i)$; cette fonction s'applique récursivement depuis la dernière image.

Cette approche permet de stocker de manière économique une séquence linéaire d'images similaires binaires. Cependant, la récupération d'une image de la séquence est assez complexe. Il faut en effet, en fonction de la place de l'image dans la séquence, appliquer récursivement les fonctions *re_overlap* ou *un_overlap* par rapport à la première ou à la dernière image, afin de pouvoir récupérer le code CBLQ de l'image et pouvoir recalculer l'image recherchée. Ce codage ne permet pas de réaliser des opérations sur les images ou de comparer différentes images non immédiatement successives. Ce point s'explique par le fait que cette approche a été créée dans le but de compresser des images de séquences vidéo. Pour appliquer une opération sur une image, il faut donc décoder l'image. De même, toutes les images non immédiatement successives doivent être décodées pour être comparées. La comparaison peut néanmoins se faire sur deux images successives, puisque le code CQBL chevauché est calculé à partir des différences entre deux images successives dans la séquence.

Les approches de compactage d'images organisées en arbres quaternaires linéaires [Lin97, TVM98] ne s'appliquent qu'aux images en noir et blanc.

4.3.3 Conclusion sur l'état de l'art

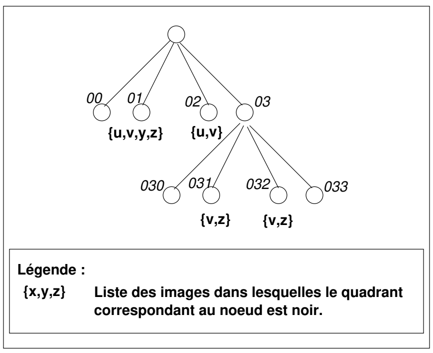


FIG. 4.9 – **La représentation des images de la figure 4.7 par un FI-Quadtree.**

L'approche de [CT91] propose d'utiliser un arbre quaternaire unique, appelé *FI-Quadtree* (ou *Full-Inverted Quadtree*), pour représenter un ensemble d'images. Cette structure est utilisée pour représenter les images binaires et permet de faire des recherches exactes ou approchées de motifs dans les images [ABH97]. Pour chaque nœud du FI-Quadtree, il existe un nœud dans au moins un arbre quaternaire représentant une image de l'ensemble. A chaque nœud de cet arbre est associé un ensemble d'identificateurs d'image. Lorsqu'un identificateur d'image apparaît dans la liste d'un nœud, le quadrant correspondant dans l'image est noir.

Exemple 4.5 La figure 4.9 illustre la représentation des images de la figure 4.7 par cette structure. Le quadrant 00 est blanc dans toutes les images, par conséquent aucun identificateur d'image ne lui est associé dans le *FI-Quadtree*. En revanche, le quadrant 01 est noir dans les images u à z, leurs identificateurs sont donc associés au nœud correspondant dans le FI-Quadtree. Le quadrant 02 est noir dans les images u, v, l'étiquette $\{u, v\}$ est donc associée au nœud 02.

Toutefois, cette approche, comme les approches basées sur un codage linéaire des arbres quaternaires, est restreinte à gérer des images en noir en blanc. En outre, le nombre d'identificateurs contenus dans une étiquette de nœud peut très vite augmenter et devenir difficile à gérer.

Il n'existe pas, à notre connaissance, d'approches permettant de minimiser l'espace de stockage des images tout en permettant à l'utilisateur d'opérer sur ces images. Dans les applications dites de vidéo à la demande (*video on demand*), les images doivent être totalement ordonnées, ce qui est le cas dans les approches [Lin97, TVM98] précédemment citées. En revanche, dans les applications pour lesquelles cette thèse apporte une contribution, les utilisateurs doivent pouvoir opérer sur des images quelconques de la base ou les comparer. Or, les approches de chevauchement facilitent seulement la comparaison d'images immédiatement successives dans la séquence.

Conclusion

L'arbre quaternaire permet de représenter des images découpées selon différents critères. Les approches de chevauchement d'arbres quaternaires, présentées dans ce chapitre, optimisent le stockage d'images similaires, généralement des images de séquences vidéo. Toutefois, les approches non restreintes à la gestion d'images binaires, ne permettent pas de répondre à tous les besoins des utilisateurs de traitement d'images. Par conséquent, nous avons proposé une structure de données optimisant le stockage et facilitant la gestion d'images similaires, dont la similarité est calculée par la distance séparant les arbres quaternaires représentant les images. Cette structure de données, l'*Arbre Quaternaire Générique*, est présentée dans le chapitre suivant.

Chapitre 5

Arbre Quaternaire Générique

Résumé

Ce chapitre décrit l'*Arbre Quaternaire Générique*. Cette structure de données offre une optimisation du stockage des images, par partage d'information entre les arbres quaternaires représentant les images. Ce partage est réalisé de sorte qu'il est possible d'appliquer des opérations spécifiques sur les images. L'idée de base est d'appliquer aux images les mécanismes de gestion de versions propres aux bases de données, et notamment ceux proposés dans l'Approche des Versions de Bases de Données. L'Arbre Quaternaire Générique, présenté dans ce chapitre, permet de ne stocker que les différences entre les arbres quaternaires représentant les images, afin d'optimiser le stockage des images. En plus de l'optimisation de l'espace occupé par les images, l'Arbre Quaternaire Générique permet d'effectuer des traitements sur les images, d'insérer ou supprimer des images dans la base, de les mettre à jour, et de les comparer. Cette structure se base sur plusieurs concepts : le partage d'information entre arbres quaternaires, la similarité entre les arbres représentant les images ou *Q-similarité*, l'*Arbre d'Images* qui organise les images dans la base et l'Arbre Quaternaire Générique, en tant que tel, qui permet de stocker les arbres quaternaires des images. Un prototype implantant l'Arbre Quaternaire Générique a été développé à l'Université Centrale du Venezuela. Proposé pour une application de traitement d'images, l'Arbre Quaternaire Générique peut être utilisé dans tous les domaines où des images similaires sont manipulées (Systèmes d'Information Géographiques, Systèmes d'Interrogation d'Images par le Contenu, etc.). Les mécanismes proposés peuvent également s'appliquer à d'autres structures (arbre R par exemple), étendant ainsi le domaine d'application de l'approche.

Introduction

Dans ce chapitre, une structure de données permettant de gérer les images est proposée. Cette structure, appelée *Arbre Quaternaire Générique* [JMR99a], stocke des images similaires organisées en arbre quaternaire, la similarité des images étant définie par la distance entre les arbres quaternaires les représentant. L'Arbre Quaternaire Générique, basé sur l'approche des Versions de Base de Données [CJ90, CJ98] (voir chapitre 2), minimise l'espace de stockage, par partage de parties communes entre images. Via cette structure, un utilisateur peut facilement choisir une ou plusieurs images dans la base de données, pour travailler avec l'image en utilisant, par exemple, un outil de traitement d'images. L'utilisateur peut également modifier une image existante dans la base, insérer ou supprimer des images, extraire des images pour construire des séquences, etc. Il s'agit par conséquent d'une approche plus générale que celles présentées dans le chapitre précédent.

L'organisation de ce chapitre est la suivante. La section 5.1 décrit les principes généraux de l'Arbre Quaternaire Générique. La section 5.2 présente les opérations applicables à l'Arbre Quaternaire Générique. La section 5.3 montre comment un ensemble d'images peut être organisé en Arbre Quaternaire Générique. Finalement, la section 5.4 décrit le prototype implantant les mécanismes de l'approche.

Dans ce chapitre, l'Arbre Quaternaire Générique est présenté sur un cas particulier d'images en noir et blanc. Ce choix de simplification a été fait dans l'unique but de faciliter la compréhension du lecteur. Cependant, l'approche proposée peut s'appliquer à tout type d'images organisées en arbre quaternaire. Un exemple d'application sur un ensemble d'images médicales est présenté à la fin du chapitre.

5.1 Principes de l'Arbre Quaternaire Générique

L'Arbre Quaternaire Générique est une nouvelle structure permettant de représenter et de gérer des images similaires dans une base de données d'images. Cette section présente les principes généraux de la structure. En 5.1.1, le concept de partage entre arbres quaternaires d'images est présenté. En 5.1.2, la notion de similarité entre images est définie. En 5.1.3, la structure d'*Arbre d'Images* est présentée. En 5.1.4, les *nœuds génériques* composant l'Arbre Quaternaire Générique sont définis. Finalement, la manière avec laquelle les images sont identifiées par le système est présentée en 5.1.5.

5.1.1 Partage entre arbres quaternaires d'image

L'Arbre Quaternaire Générique est basé sur le principe de partage de régions (ou quadrants) entre images. Soit $\mathcal{I}_m$, un ensemble d'images. Si un quadrant q a la même valeur dans un ensemble d'images $\mathcal{I'}_m \subset \mathcal{I}m$, cette valeur n'est stockée qu'une seule fois dans la base et est associée à l'ensemble des identificateurs des images de $\mathcal{I'}_m$. Dans ce cas, on parle de *partage explicite*, parce que l'identificateur de chaque image partageant cette valeur apparaît explicitement dans la liste des images associées à cette valeur. Par exemple, dans les images de la figure 3.1 (voir page 29), il serait intéressant de stocker la région 011 de l'image originale, la même région dans l'image après traitement et de ne stocker qu'une seule fois toutes les autres régions puisque le traitement ne les a pas affectées.

Si les images de l'ensemble $\mathcal{I}_m$ sont organisées en arborescence, chaque image, excepté la racine de l'arbre, a une mère unique et un nombre indéfini d'images filles. Par conséquent, la règle de *partage implicite* suivante peut être introduite : *"Excepté lorsque l'identificateur d'une image i est explicitement associé avec une autre valeur v, l'image i partage implicitement la valeur associée à son image mère."*.

Si l'arbre organisant les images est stocké, cette règle de partage implicite permet une représentation compacte de l'ensemble d'images, en particulier lorsqu'un grand nombre d'images partagent des valeurs de quadrants dans plusieurs branches de l'arbre.

5.1.2 Similarité entre images

Les images sont regroupées, dans la base de données, en fonction d'une distance de similarité entre les arbres quaternaires qui les représentent. Cette distance, appelée *Q-similarité*, est proposée afin d'optimiser le stockage des images dans la base. La distance de Q-similarité entre deux images est définie par le nombre de nœuds différents (de même identificateur et de valeur différente) dans les deux arbres quaternaires représentant les images, divisé par le nombre d'identificateurs de nœud (sans doublon) apparaissant dans l'union des nœuds des arbres quaternaires des images.

De manière plus formelle, on note $S(i,i')$ l'ensemble des nœuds différents entre les arbres quaternaires des images i et i'. On note $U(i,i')$ l'ensemble (sans doublon) des identificateurs de nœud apparaissant dans les arbres quaternaires des images i et i'. On note $\mid S(i,i') \mid$ (resp. $\mid U(i,i') \mid$) le nombre d'éléments de $S(i,i')$ (resp. de $U(i,i')$). La distance de Q-similarité des images i et i', notée $d(i,i')$, est calculée par l'équation suivante :

$$d(i,i') = \frac{\mid S(i,i') \mid}{\mid U(i,i') \mid} = \frac{\text{Nombre de nœuds différents}}{\text{Total des identificateurs de nœud (sans doublon)}}$$

- $d(i,i') \in [0,1]$, c'est une distance normalisée (voir démonstration en annexe A1), c'est-à-dire : $d(i,i) = 0$, $d(i,i') = d(i',i)$ et $d(i,i') \leq d(i,j) + d(j,i')$ (inégalité triangulaire).
- $d(i,i') = 0 \iff$ les arbres quaternaires des images i et i' partagent tous leurs nœuds.
- $d(i,i') = 1 \iff$ les arbres quaternaires des images i et i' ne partagent aucun nœud.

On définit un *coefficient de Q-similarité* de deux images i et i', égal à $1 - d(i,i')$. Ce coefficient indique la proportion de nœuds communs (de même identificateur et de même valeur) entre les arbres quaternaires des images i et i'. Deux images sont considérées comme ε-Q-similaires si leurs arbres quaternaires ont une proportion ε de nœuds communs.

Exemple 5.1 La distance de Q-similarité des images a et b, dont les arbres quaternaires sont représentés sur la figure 5.1, est égale à :
$d(a,b) = \frac{|S(a,b)|}{|U(a,b)|} = \frac{5}{9}$

- $\mid U(a,b) \mid = 9$, car les nœuds 0, $0x$ et $03x$, $x \in [0,3]$ apparaissent dans l'union des identificateurs des nœuds des arbres quaternaires des images a et b (voir figure 5.1).

- $\mid S(a,b) \mid = 5$, car les nœuds 03 et $03x$, $x \in [0,3]$ sont différents dans les deux arbres quaternaires.

Les images a et b sont 4/9-Q-similaires.

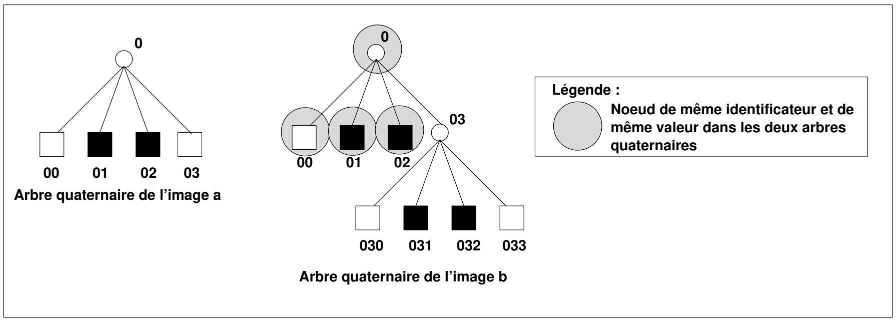


FIG. 5.1 – **Le calcul du coefficient de Q-similarité des images** a **et** b.

5.1.3 Arbre d'Images

En conséquence de la règle de partage implicite énoncée dans la section 5.1.1, les images représentées par un Arbre Quaternaire Générique sont organisées à l'aide d'une structure arborescente particulière, l'*Arbre d'Images*. Lorsqu'une nouvelle image est insérée dans l'Arbre d'Images, elle est insérée comme fille de l'image dont elle est la plus Q-similaire, c'est-à-dire dont la distance entre l'arbre quaternaire associé et celui de l'image à insérer est la plus proche de 0.

L'Arbre d'Images est utilisé pour optimiser l'espace de stockage en permettant le partage implicite entre images. La Q-Similarité permet de calculer l'économie réalisée.

Exemple 5.2 La figure 5.2 présente un Arbre d'Images organisant les images a, b, c et d. On suppose que l'image a est une image originale sur laquelle ont été appliqués des traitements dont les images b et c ont résulté. L'image d correspond au résultat de l'image b après traitement. L'utilisateur a dans ce cas, aidé à l'organisation des images.

La section 5.3 montre comment un ensemble d'images peut être organisé en Arbre d'Images. L'insertion d'une image dans l'Arbre d'Images est présentée en 5.2.3. Dans le prototype implantant les mécanismes de l'Arbre Quaternaire Générique (voir section 5.4), l'organisation des images dans l'Arbre d'Images se fait de manière interactive avec l'utilisateur.

5.1.4 Nœuds génériques

La représentation et le stockage d'un ensemble d'images similaires sont effectués dans un *Arbre Quaternaire Générique*, dont les nœuds sont appelés *nœuds génériques*. Pour chaque nœud apparaissant dans l'arbre quaternaire d'une image, il existe un nœud générique ayant le même identificateur (voir chapitre 4, section 4.2.1) dans l'Arbre Quaternaire Générique. Un nœud générique n représente tous les nœuds n des arbres quaternaires des images de la base. Il contient

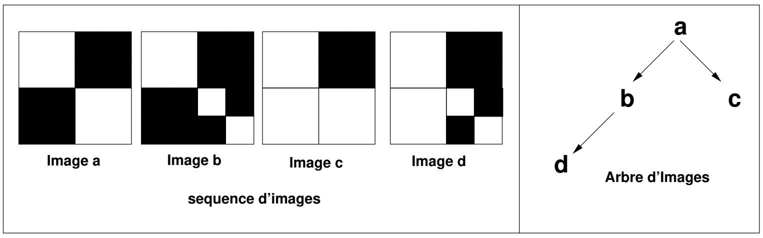


FIG. 5.2 – **Les images sont organisées en Arbre d'Images.**

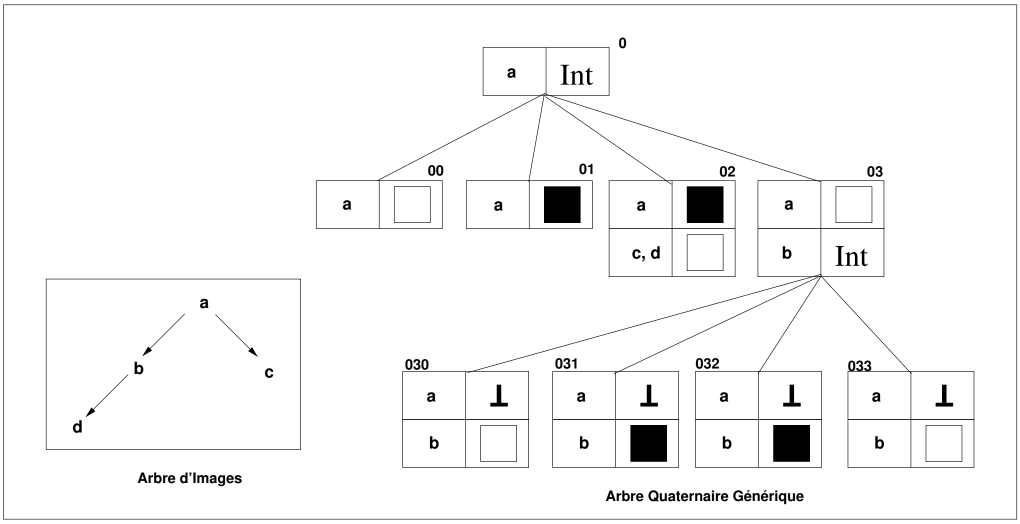


FIG. 5.3 – **L'Arbre Quaternaire Générique des images de la figure 5.2.**

toute l'information nécessaire pour recomposer la valeur du nœud de même identification dans chaque arbre quaternaire.

La valeur d'un nœud est soit $\perp$ qui signifie que le nœud *n'existe pas*, soit Int qui signifie qu'il est interne (il a quatre fils), soit *noir* (un carré de couleur noire) s'il est une feuille noire, ou *blanc* (un carré de couleur blanche) s'il est une feuille blanche.

Chaque nœud générique peut être vu comme un tableau ayant une ou plusieurs lignes. Chaque ligne l de nœud générique n contient une liste d'identificateurs d'image et une valeur v de nœud d'arbre quaternaire. La valeur v signifie que tous les nœuds identifiés par n ont pour valeur v dans les arbres quaternaires des images dont l'identificateur i apparaît dans la liste. De plus, par application de la règle de partage implicite énoncée en 5.1.1, on déduit que les nœuds n des arbres quaternaires de toutes les images descendantes des images de la liste, dans l'Arbre d'Images, partagent implicitement cette même valeur, excepté lorsqu'un identificateur d'image descendante apparaît explicitement dans une autre ligne du nœud générique.

Dans les explications qui suivent, l'arbre quaternaire d'une image i désigne l'arbre quaternaire qui serait associé à i si celle-ci était considérée isolément, c'est-à-dire en dehors de l'arbre générique.

Exemple 5.3 La figure 5.3 représente l'Arbre d'Images et l'Arbre Quaternaire Générique des images de la figure 5.2. Le nœud générique 0 contient une seule ligne de valeur Int : elle signifie que les nœuds 0 sont internes dans les arbres quaternaires de toutes les images de l'ensemble. En effet, cette valeur est explicitement associée à l'image a. Elle est donc implicitement partagée par les arbres quaternaires des images b, c et d, puisqu'elles sont descendantes de l'image a dans l'Arbre d'Images et qu'elles n'apparaissent dans aucune autre ligne du nœud générique 0. Le nœud générique 02 contient deux lignes. La première ligne signifie de manière explicite que le nœud 02 est noir dans l'arbre quaternaire de l'image a, et de manière implicite qu'il est noir dans l'arbre quaternaire de l'image b. Cette valeur n'est pas partagée par les arbres quaternaires des images c et d, puisque leurs identificateurs apparaissent dans la deuxième ligne du nœud générique 02, de valeur $blanc$. La valeur $\perp$ des nœuds $03x$, $x \in [0;3]$, signifie que ces nœuds n'existent pas dans les arbres quaternaires des images a (de manière explicite) et c (par partage implicite).

5.1.5 Identification interne des images

Les utilisateurs finaux peuvent identifier les images selon des critères qui leur sont propres ou qui sont propres à l'application. L'identification externe des images (a, b, c et d dans l'exemple) est vue par les utilisateurs et est transformée, par le système, en identification interne. Cette identification interne optimise la recherche dans l'Arbre Quaternaire Générique. Pour optimiser cette recherche, l'identification interne est construite de telle sorte que tous les ancêtres d'une image dans l'Arbre d'Images puissent être déduits de l'identificateur interne de l'image. Ce type de codage est utilisé dans [CJ90, CJ98] et [KU95].

Ce point est important dans le sens où l'implantation physique des images similaires est complètement indépendante de la manière avec laquelle les images sont vues par les utilisateurs. Plus précisément, les utilisateurs peuvent extraire les images et leurs arbres quaternaires de l'Arbre Quaternaire Générique et les ordonner comme ils veulent. Ils peuvent par exemple construire des séquences d'images en ordonnant les images selon un critère donné.

L'organisation d'un ensemble d'images en Arbre Quaternaire Générique est décrite dans la section 5.3.

5.2 Opérations

Cette section présente plusieurs opérations applicables aux images stockées dans l'Arbre Quaternaire Générique : la lecture d'une image décrite en 5.2.1, la modification d'une image décrite en 5.2.2, l'insertion / la suppression d'une image dans la base présentées en 5.2.3 et 5.2.4. La réorganisation de l'Arbre d'Images est présentée en 5.2.5. D'autres opérations telles que la comparaison d'images, l'union, l'intersection, etc., sont présentées en 5.2.6.

5.2.1 Lecture d'une image

La lecture d'une image de l'Arbre Quaternaire Générique est faite depuis sa racine, le nœud générique 0, et la valeur de ce nœud dans l'arbre quaternaire de l'image i est déterminé grâce à la règle de partage implicite énoncée en 5.1.1. Si la valeur d'un nœud n est Int pour l'arbre quaternaire de l'image i, alors le nœud est interne dans l'arbre quaternaire de i et les nœuds génériques, fils de n, sont lus. En revanche, si le nœud n est feuille dans l'arbre quaternaire de l'image i, alors la valeur du nœud générique n, associée à i, correspond à la valeur du nœud n dans l'arbre quaternaire de i.

Exemple 5.4 Les étapes qui suivent permettent de lire l'image c stockée dans l'Arbre Quaternaire Générique de la figure 5.3 :

1. Le nœud générique 0, racine de l'Arbre Quaternaire Générique, est lu. Comme l'identificateur de l'image c n'apparaît pas dans le nœud générique, la règle de partage implicite permet de déduire que l'image c partage la valeur associée à son ancêtre le plus proche dans l'Arbre d'Images, l'image a. La valeur du nœud 0 associée à l'image a est Int. Par conséquent, le nœud 0 est interne dans l'arbre quaternaire de l'image a, et par partage implicite on déduit qu'il est interne dans l'arbre quaternaire de l'image c. Les nœuds fils de 0 vont être lus lors les trois étapes suivantes.

2. La règle de partage implicite permet de déduire que les nœuds 00 et 01 sont respectivement $blanc$ et $noir$ dans l'arbre quaternaire de l'image c.

3. La valeur $blanc$ est explicitement associée à l'image c dans le nœud générique 02, on déduit que ce nœud est une feuille blanche dans l'arbre quaternaire de l'image c.

4. Finalement, l'image c et son image mère a partagent la valeur $blanc$ du nœud générique 03.

5.2.2 Modification d'une image

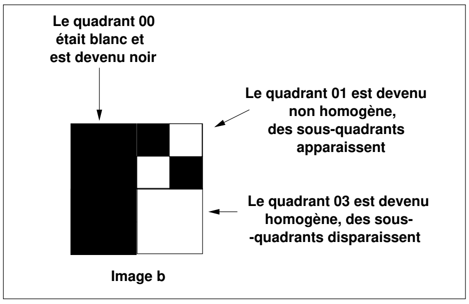


FIG. 5.4 – **La modification de l'image b.**

La modification d'une image entraîne la modification de son arbre quaternaire, et par conséquent celle de l'Arbre Quaternaire Générique. Il existe deux types de modification, illustrés par les modification apportées à l'image b sur la figure 5.4 par rapport à l'image b de la figure 5.2.

1. La modification préserve la structure de l'arbre quaternaire représentant l'image et seules les valeurs des nœuds feuilles sont modifiées. Par exemple, dans la figure 5.4, le quadrant 00

de l'image b a été modifié. Sa couleur est passée de *blanc* (voir figure 5.2) à *noir* (voir figure 5.4). Par conséquent, le nœud 00 de l'arbre quaternaire de l'image b est modifié.

2. La modification change la structure de l'arbre quaternaire de l'image :

   (a) Des nœuds peuvent disparaître parce qu'un nœud non homogène est devenu homogène. Par exemple, dans la figure 5.4, les nœuds $03x$, $x \in [0,3]$, disparaissent de l'arbre quaternaire de l'image b parce que le quadrant 03 est devenu homogène : *blanc* (voir figure 5.4).

   (b) Des nœuds peuvent apparaître parce qu'un quadrant homogène est devenu non homogène. Par exemple, les nœuds $01x$, $x \in [0,3]$, apparaissent dans l'arbre quaternaire de l'image b parce que le quadrant 01 est devenu non homogène (voir figure 5.4).

Tous ces cas peuvent apparaître simultanément sur une même image originale.

a) Modification de la valeur d'un nœud d'arbre quaternaire

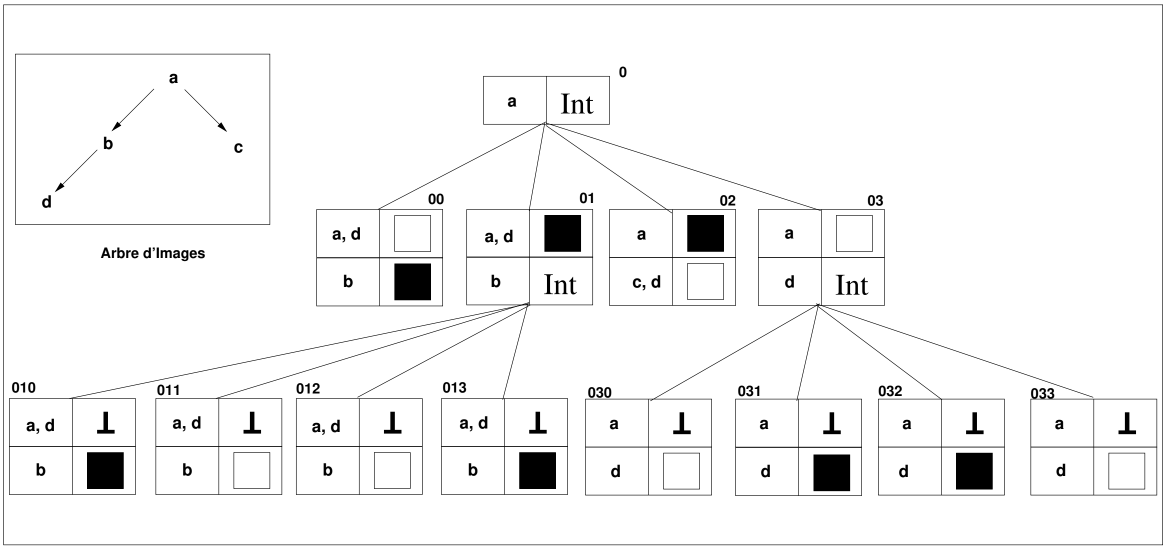


FIG. 5.5 – **L'Arbre Quaternaire Générique après la modification de l'image b.**

La modification de la valeur d'un nœud n d'arbre quaternaire, représentant une image i, modifie le nœud générique n dans l'Arbre Quaternaire Générique. La nouvelle valeur, notée *new*, de n doit être explicitement ou implicitement associée à i dans le nœud générique. Cependant, pour ne pas modifier les éventuelles images descendantes de i, il faut supprimer le partage implicite entre i et ses filles. Il faut donc s'assurer que toute image fille de i est explicitement associée à une valeur de n et, si ce n'est pas le cas, l'associer explicitement à la valeur *old* qu'elle partageait implicitement avec i avant la modification.

L'algorithme de modification de la valeur *old* du nœud n dans l'arbre quaternaire de l'image i est le suivant :

> Soit n un nœud générique,
> Soit i l'image modifiée,
> Soit *new* la nouvelle valeur du nœud n dans l'arbre quaternaire de i.
> Soit *old* la valeur du nœud n dans l'arbre quaternaire de l'image i, avant modification : la valeur *old* est associée à i dans le nœud générique n.

1. Si i est la seule image associée explicitement à la valeur *old* dans le nœud générique n, alors *old* est remplacée par *new*.
2. Si i n'est pas la seule image associée explicitement à la valeur *old* dans le nœud générique n, parce que d'autres images partagent implicitement ou explicitement cette valeur, alors l'association entre i et la valeur *old* est supprimée et :
   (a) Si la valeur *new* existe dans le nœud générique n, alors deux cas se présentent :
   (α) si l'image mère de i, dans l'Arbre d'Images, est implicitement ou explicitement associée à la valeur *new* de n, alors i partage implicitement cette valeur avec son image mère.
   (β) Si l'image mère de i n'est pas implicitement ou explicitement associée à la valeur *new*, alors il faut ajouter l'indentificateur de i dans la ligne de valeur *new* du nœud générique n.
   (b) Si la valeur *new* n'existe pas dans le nœud générique n, alors une nouvelle ligne l est ajoutée dans le nœud générique n ; cette ligne associe l'image i à la valeur *new*.
3. Pour toutes les images i', filles de l'image i dans l'Arbre d'Images :

   Si i' n'apparaît dans aucune ligne du nœud générique n et donc partageait implicitement la valeur *old* avec l'image i, alors i' est explicitement associée à la valeur *old* dans le nœud générique n.

Exemple 5.5 Après modification de l'image b (voir figure 5.4), la valeur *noir* du nœud 00 est explicitement associée à l'image b dans la figure 5.5. L'image d, fille de l'image b dans l'Arbre d'Images, est explicitement associée à la valeur *blanc* qu'avait le nœud 00 dans l'arbre quaternaire de l'image b avant modification.

b) Suppression de nœuds d'arbre quaternaire

Lorsque la modification d'une image i entraîne la suppression de nœuds (au minimum quatre, notés $n0$, $n1$, $n2$ et $n3$) dans l'arbre quaternaire représentant l'image, les nœuds génériques correspondants doivent être mis à jour de sorte que l'image i soit associée à la valeur $\perp$ des nœuds, $\perp$ signifiant *n'existe pas*. Comme dans la section précédente, pour ne pas modifier les images descendantes de i, il faut supprimer tout éventuel partage implicite entre i et ses filles. Cette opération correspond à une modification de valeur de nœud. Si un nœud générique ne contient la valeur est $\perp$ que pour l'image modifiée, il peut être supprimé de la base et donc de l'Arbre Quaternaire Générique. La suppression de nœuds frères $n0$, $n1$, $n2$ et $n3$ dans un arbre quaternaire entraîne également la mise à jour du nœud père n : ce dernier n'est plus interne et devient feuille. Le nœud générique n doit par conséquent être mis à jour.

Exemple 5.6 Le nœud 03, qui était interne (voir figure 5.3) dans l'arbre quaternaire de l'image b devient *blanc* (voir figure 5.5). Dans l'Arbre Quaternaire Générique, l'identificateur de l'image b est supprimé de la liste d'identificateurs d'images associés à la valeur *Int* du nœud générique 03. L'image d, fille de b dans l'Arbre d'Images, est explicitement associée à cette valeur. Puis, l'image b est associée à la valeur *blanc* du nœud générique 03. Comme cette valeur existe et est déjà associée à l'image a, mère de b dans l'Arbre d'Images, l'image b partage implicitement cette valeur avec l'image a. Après la modification du nœud 03, les nœuds $03x$, $x \in [0, 3]$, sont supprimés de l'arbre quaternaire de l'image b. Par conséquent, l'image b partage implicitement, pour ces nœuds, la valeur $\perp$ avec l'image a, cette dernière étant explicitement associée à $\perp$ et étant mère de b. L'image d, fille de l'image b, est explicitement associée aux valeurs *noir* et *blanc* qu'avaient les nœuds $03x$ dans l'arbre quaternaire de l'image b avant modification.

c) Création de nœuds d'arbre quaternaire

De nouveaux nœuds (au minimum quatre, notés $n0$, $n1$, $n2$ et $n3$) peuvent être ajoutés dans un arbre quaternaire suite à la modification d'une image i. Cette opération modifie le nœud générique père n de sorte que la valeur *Int* (nœud interne) soit associée à l'image i, et seulement à i, en supprimant l'éventuel partage implicite entre i et ses descendants. Pour chaque nouveau nœud nx d'arbre quaternaire, $x \in [0, 3]$, un nœud générique nx correspondant est créé dans l'Arbre Quaternaire Générique, s'il n'existe pas déjà. Tout d'abord, le nœud générique nx est créé vide : la valeur $\perp$ est associée à l'image racine de l'Arbre d'Images. Puis le nœud nx est mis à jour de sorte que l'image i soit associée, dans le nœud générique nx, à la valeur que le nœud nx a dans l'arbre quaternaire de l'image i (voir sous-section a)).

Exemple 5.7 Cette situation apparaît dans la figure 5.5. Les nœuds $01x$, $x \in [0, 3]$, sont créés dans l'arbre quaternaire de l'image b, car le quadrant 01 de l'image b qui était homogène (voir figure 5.2) ne l'est plus (voir figure 5.4). Comme les nœuds génériques correspondants n'existent pas, ils sont créés dans l'Arbre Quaternaire Générique avec la valeur $\perp$ associée explicitement à l'image a, racine de l'Arbre d'Images. Puis ces nœuds sont mis à jour de sorte que la valeur *blanc* ou *noir* soit associée explicitement à l'image b. L'image d, fille de b dans l'Arbre d'Images, est associée à la valeur $\perp$ de ces nœuds (le partage implicite entre les images b et d est rompu). Une nouvelle ligne est ajoutée au nœud 01 dans laquelle la valeur *Int* est associée à l'image b, comme il est décrit dans la sous-section a). L'image d, fille de l'image b, est explicitement associée à la valeur qu'avait le nœud 01 dans l'arbre quaternaire de l'image b avant modification, c'est-à-dire *noir*.

d) Autres modifications

A travers les exemples présentés précédemment, il est clair que par des modifications similaires sur les nœuds génériques, il est possible :

- de modifier une partie d'image par copie d'un morceau d'une image existante. La copie revient à mettre à jour les nœuds d'arbre quaternaire correspondants à la partie copiée ;

- de modifier une partie d'image par copie en profondeur d'un sous arbre quaternaire. Cette opération revient à modifier plusieurs nœuds d'arbre quaternaire ;
- de répercuter des modifications dans plusieurs images simultanément. La répercussion des mises à jour la plus facile à mettre en œuvre est la propagation des modifications d'une image i à toutes ses images descendantes dans l'Arbre d'Images. En effet, cette opération consiste à modifier les valeurs de nœud associées à l'image i, et à supprimer les identificateurs des images descendantes de i de tous les nœuds génériques modifiés. Par conséquent, les modifications appliquées aux nœuds de l'arbre quaternaire de l'image i sont répercutées aux nœuds des arbres quaternaires des images filles, petites-filles etc., de i, par le mécanisme de partage implicite des valeurs.

e) Conséquences des modifications

La modification d'une image i peut entraîner une diminution du coefficient de Q-similarité entre i et ses images descendantes. Il est possible, dans ce cas, de réorganiser l'Arbre d'Images pour réorganiser le partage entre les images stockées dans l'Arbre Quaternaire Générique. Cette opération est décrite dans la section 5.2.5. La réorganisation de l'Arbre d'Images ne change rien au partage explicite entre les images mais permet, en revanche, d'optimiser le partage implicite. En effet, le partage explicite des valeurs de nœuds consiste à ne stocker qu'une seule fois dans la base des valeurs communes à plusieurs arbres quaternaires. Le partage implicite diminue le nombre d'identificateurs d'images apparaissant dans les lignes de nœud générique. Par conséquent, une réorganisation de l'Arbre d'Images optimise le partage implicite.

5.2.3 Insertion d'image

Il existe deux modes d'insertion d'image dans une base de données gérée à l'aide d'un Arbre Quaternaire Générique. Le premier mode consiste à créer une nouvelle image par modification d'une image existant dans la base, sans suppression de cette dernière. L'autre consiste à insérer une nouvelle image, créée en dehors de la base, par un outil quelconque de traitement d'images.

a) Insertion par copie d'image

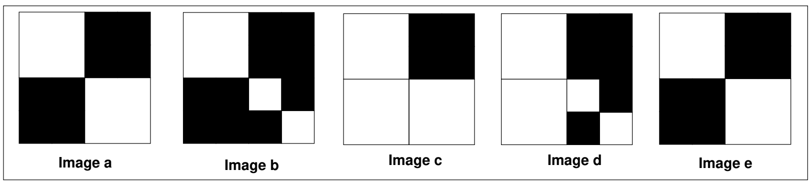


FIG. 5.6 – **La création de l'image e comme copie de l'image a.**

Dans le premier cas, une nouvelle image $x.i$ est insérée, dans l'Arbre d'Images, comme fille d'une image x existant dans la base. Seul l'Arbre d'Images est modifié par ajout de l'image $x.i$ comme $i$ème fille de l'image x. L'Arbre Quaternaire Générique n'est pas modifié par cette insertion. La nouvelle image $x.i$ partage par conséquent toutes les valeurs de nœud associées à son image

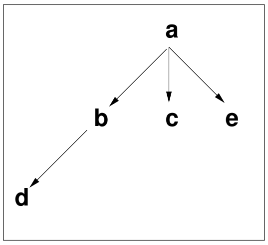


FIG. 5.7 – **La modification de l'Arbre d'Image après insertion de l'image** e.

mère x. Cette opération est donc extrêmement rapide. L'utilisateur peut, par la suite, modifier l'image insérée comme il est indiqué dans la section 5.2.2.

Exemple 5.8 La figure 5.6 illustre l'insertion de l'image e comme fille de a dans l'Arbre d'Images. Les deux images a et e sont identiques, suite à cette insertion (e est une copie de a). La figure 5.7 représente la mise à jour de l'Arbre d'Images. L'Arbre Quaternaire Générique reste identique à celui représenté sur la figure 5.5.

b) Insertion d'une image extérieure

TAB. 5.1 – **Tableau de partage d'une image** i **à insérer avec les images de la base**

[a]	$\mathbf{i}_1$	$\mathbf{i}_2$	...	$\mathbf{i}_k$	...
(n,v)	c_1	c_2	...	c_k	...
...					

[a] La première colonne du tableau de partage contient des couples (*identificateur de nœud, valeur*). La première ligne contient des identificateurs d'images. Puis, chaque case du tableau associe un coefficient c_k, de valeur 0 ou 1, à un couple (*identificateur de nœud, valeur*) et à un identificateur d'image i_k.

Dans le deuxième cas, une nouvelle image extérieure est insérée dans la base. Il peut s'agir d'un fichier extérieur ou d'une image extraite de la base, modifiée par un outil de traitement d'images, et à nouveau insérée dans la base après modification. Par conséquent, l'arbre quaternaire de la nouvelle image doit être inséré dans l'Arbre Quaternaire Générique. L'Arbre Quaternaire Générique est proposé dans l'optique de minimiser l'espace de stockage des images de la base, par partage des valeurs de nœuds d'arbre quaternaire entre images mères et filles. Il convient, par conséquent, d'insérer la nouvelle image i, dans l'Arbre d'Images, comme fille d'une image j telle que la distance de Q-similarité (voir section 5.1.2) entre les images i et j soit la plus proche de 0.

L'idée générale de l'algorithme est de chercher, dans la base, l'image j dont l'arbre quaternaire a le plus de nœuds communs, c'est-à-dire de même identificateur et de même valeur, avec l'arbre quaternaire de l'image i à insérer. L'image i est par conséquent organisée en arbre quaternaire. Un *tableau de partage* est construit. Ce tableau est composé d'autant de lignes que l'arbre quaternaire de l'image i a de nœuds, et d'autant de colonnes qu'il y a d'images dans la base (voir tableau 5.1). Chaque élément du tableau est un coefficient c de valeur 0 ou 1. Si le nœud n a pour valeur v

dans l'arbre quaternaire de l'image i' alors le coefficient c, situé à la colonne i' et à la ligne (n, v), a pour valeur 1. A l'inverse, si le nœud n a une valeur différente de v dans l'arbre quaternaire de i', alors le coefficient c a pour valeur 0. Puis, la somme des coefficients de chaque colonne est calculée. L'image i est placée, dans l'Arbre d'Images, comme fille de l'image j dont la somme des coefficients est la plus élevée. Si plusieurs images sont candidates, on choisit de préférence celle qui est située la moins profondément dans l'Arbre d'Images. Suite à cette insertion dans l'Arbre d'Images, l'image i est identique à son image mère j. Elle est par conséquent mise à jour : chaque nœud générique n pour lequel le coefficient associé à j est 0 dans le tableau de partage, est modifié de sorte que i soit associée à la valeur v qu'a le nœud n dans son arbre quaternaire.

Exemple 5.9 L'image f (voir figure 5.8), créée à l'extérieure de la base, est insérée dans l'Arbre Quaternaire Générique de la figure 5.3 (voir page 53) :

1. L'arbre quaternaire de l'image f est construit (voir figure 5.8). Le tableau de partage de f est construit (voir tableau 5.2). L'image d a la somme des coefficients la plus élevée. Par conséquent, f est placée comme fille de l'image d dans l'Arbre d'Images.

2. Suite à l'insertion de f dans l'Arbre d'Images, l'image f est identique à l'image d. Par conséquent, les nœuds $03x$, $x \in [0, 3]$ sont mis à jour dans l'image f de sorte que f soit associée, dans l'Arbre Quaternaire Générique, à la valeur de ses nœuds dans l'arbre quaternaire de f. L'Arbre Quaternaire Générique de la figure 5.9 représente l'Arbre Quaternaire Générique après insertion de l'image extérieure f.

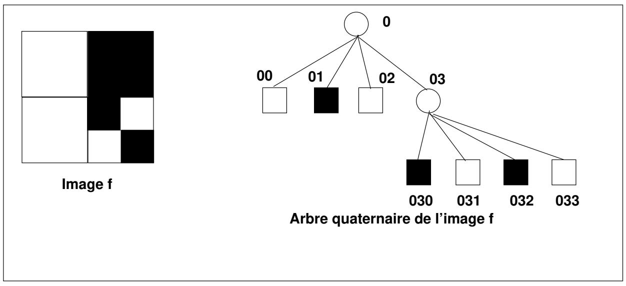


FIG. 5.8 – **Une image f et sa représentation en arbre quaternaire.**

5.2.4 Suppression d'image

Il existe plusieurs mécanismes de suppression d'une image dans l'Arbre Quaternaire Générique : la suppression logique et la suppression physique. Plusieurs variantes de la suppression physique sont également envisageables.

a) Suppression logique

La suppression logique d'une image i consiste à cacher l'image aux utilisateurs. Lorsqu'une image i est supprimée logiquement de la base, l'identificateur i n'apparaît plus aux utilisateurs finaux. Par conséquent, l'image i n'apparaît plus dans l'ensemble d'images. Les utilisateurs ne

Tab. 5.2 – **Tableau de partage de l'image f avec les images a, b, c et d**

	a	b	c	d
$(0, Int)$	1	1	1	1
$(00, B)$	1	1	1	1
$(01, N)$	1	1	1	1
$(02, B)$	0	0	1	1
$(03, Int)$	0	1	0	1
$(030, N)$	0	0	0	0
$(031, B)$	0	0	0	0
$(032, N)$	0	0	0	0
$(033, B)$	0	0	0	0
Total	3	4	4	5

peuvent plus accéder à l'image supprimée et l'image ne peut plus avoir de nouvelles images filles.

Pour ne pas modifier les images filles de i, existant dans la base avant la suppression de i et qui partagent implicitement des valeurs avec i, la structure de l'Arbre Quaternaire Générique et celle de l'Arbre d'Images restent inchangées. Seul un *indicateur de suppression* de l'image i est positionné dans l'Arbre d'Images. Les références à l'image supprimée existent toujours dans les nœuds génériques, mais elles demeurent invisibles à l'utilisateur.

b) Suppressions physiques

La suppression physique d'une image i implique une modification de l'Arbre Quaternaire Générique et de l'Arbre d'Images. Cette suppression peut être réalisée de trois manières :

1. Une première forme de suppression revient à supprimer les valeurs des nœuds de l'arbre quaternaire de l'image i dans les nœuds génériques correspondants, lorsqu'elles ne sont explicitement partagées à aucune autre image. Cette opération est avantageuse lorsque les valeurs à supprimer occupent beaucoup de place.

2. La deuxième forme consiste à supprimer toutes les références implicites ou explicites à l'image i dans l'Arbre Quaternaire Générique. Cette opération revient à associer explicitement les valeurs des nœuds aux images filles, lorsque le partage est implicite entre l'image et ses filles et à supprimer l'identificateur i dans tous les nœuds génériques. En revanche, l'Arbre d'Images n'est pas modifié par cette suppression, seul un indicateur de suppression de l'image i est placé dans l'arbre. Cette suppression, qui nécessite l'examen de tous les nœuds de l'Arbre Quaternaire Générique concernés par l'image supprimée, est donc une opération plus lourde que la suppression logique.

3. La troisième forme consiste à appliquer la suppression décrite en (2) en modifiant, cette fois-ci, l'Arbre d'Images, c'est-à-dire en supprimant l'identificateur i, avec les conséquences que cela entraîne au niveau des identificateurs internes des images. Cette suppression est généralement envisagée en cas d'archivage d'une partie (ex. sous-arbre) des images stockées dans l'Arbre Quaternaire Générique.

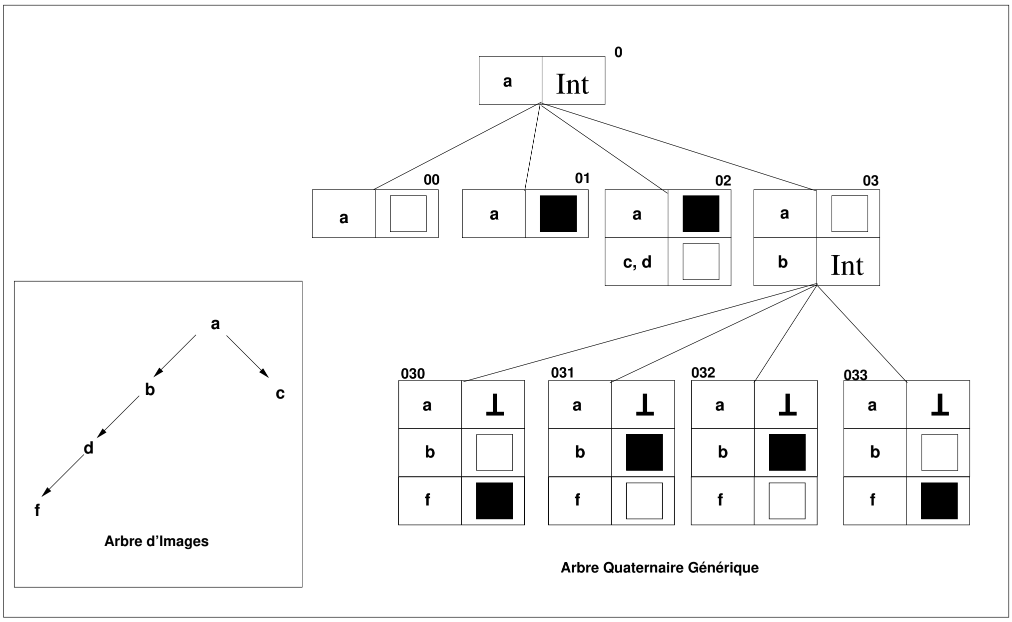


FIG. 5.9 – **La modification de la base après l'insertion de l'image extérieure *f*.**

Un exemple de suppression physique d'image est présentée dans la section suivante.

5.2.5 Réorganisation de l'Arbre d'Images

La modification d'une image peut entraîner une forte diminution du coefficient de Q-similarité entre une image et ses images descendantes. Par exemple, le coefficient [1] de Q-similarité de l'image b et de sa fille d, après modification de l'image b (voir figure 5.4 p. 55), passe de $\frac{8}{9}$ à 0, ce qui signifie qu'il n'y a plus aucun partage entre les image b et d.

Il est donc intéressant, dans le cas de mises à jour conséquentes entraînant une forte diminution du coefficient de Q-similarité, de pouvoir réorganiser l'Arbre d'Images et donc le partage entre les images. Cette réorganisation peut se faire par une série de suppressions (voir section 5.2.4) et d'insertions (voir section 5.2.3) d'images. L'image modifiée entraînant la diminution du coefficient de Q-similarité avec ses filles, notée i, doit être supprimée, puis réinsérée dans l'Arbre Quaternaire Générique, de sorte qu'elle soit placée comme fille de l'image avec laquelle la distance de Q-similarité est la plus proche de 0.

Parmi les différents types de suppression présentés dans la section 5.2.4, la suppression la plus judicieuse pour ce genre d'opération consiste (1) à éliminer l'identificateur de i des nœuds génériques pour lesquels il n'y a aucun partage implicite entre i et ses images descendantes et (2) à supprimer le partage implicite entre l'image i et ses filles. Cette suppression reprend les principes de la suppression présentée en 5.2.4 (voir item 2), à l'exception de la suppression des valeurs. En effet, ces valeurs vont à nouveau être associées à i lors de sa réinsertion, il est donc inutile de les

1. Respectivement, la Q-similarité passe de $\frac{1}{9}$ à 1.

supprimer. L'image i n'est pas supprimée de l'Arbre d'Images ; un indicateur de suppression est associé à l'identificateur i dans l'Arbre d'Images. L'image i est, par la suite, réinsérée dans l'Arbre d'Images et un nouvel identificateur interne lui est attribué.

Lorsque les images filles de i sont feuilles dans l'Arbre d'Images, elles peuvent être supprimées puis réinsérées dans la base. Il est même possible, dans ce cas, d'envisager plutôt la suppression et la réinsertion des images filles que celles de i, dans l'optique où l'optimisation du partage implicite risque d'être plus conséquente que lorsque l'image i est déplacée dans l'Arbre d'Images.

Exemple 5.10 Suite à la modification de l'image b (voir figure 5.4 p. 55), l'image b et son image fille d sont zéro-Q-similaires. Par conséquent, un réorganisation de l'Arbre d'Images s'impose pour optimiser le partage entre les images stockées dans l'Arbre Quaternaire Générique de la figure 5.5 (page 56). Plusieurs optiques sont envisageables :

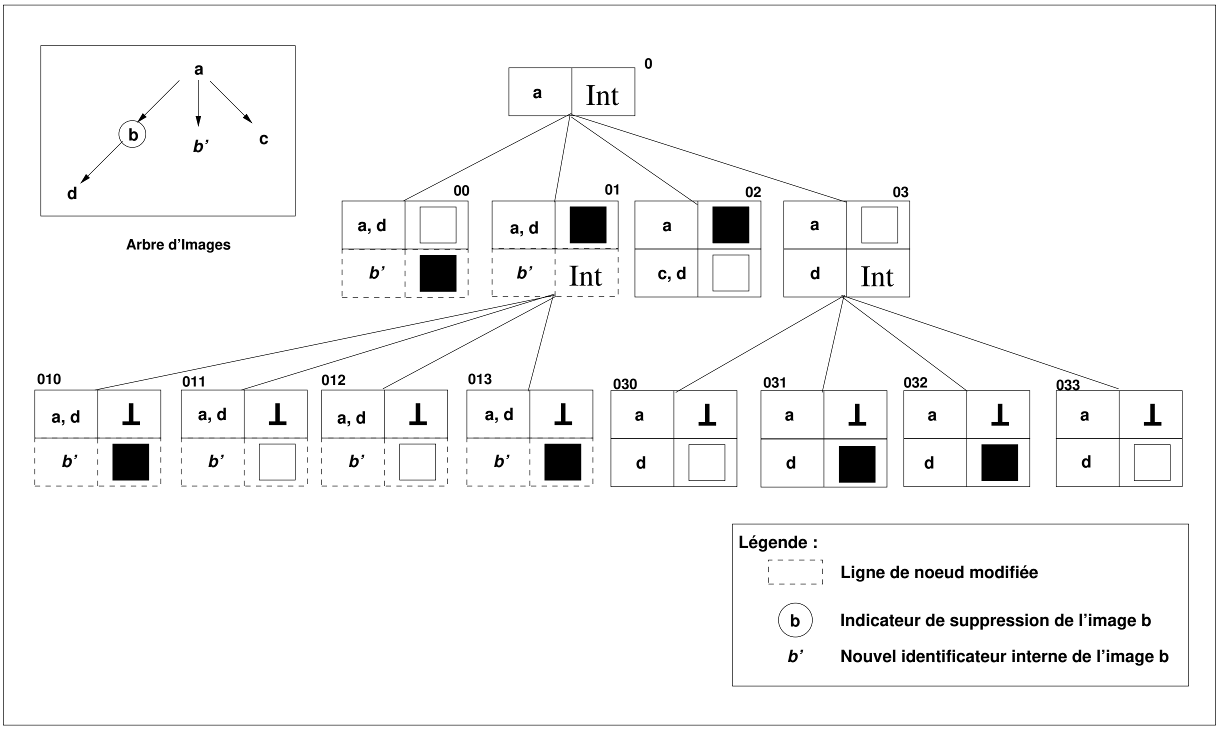


FIG. 5.10 – **La réorganisation de l'Arbre d'Images et de l'Arbre Quaternaire Générique par suppression et réinsertion de l'image** b.

1. L'image b est supprimée. Son identificateur disparaît de l'Arbre Quaternaire Générique et un indicateur de suppression est ajouté à b dans l'Arbre d'Images. Puis l'image b est à nouveau insérée dans l'Arbre Quaternaire Générique (voir section 5.2.3) comme fille de l'image a dont la distance de Q-similarité est la plus proche de 0. Un nouvel identificateur interne, b', lui est attribué. La figure 5.10 représente l'Arbre Quaternaire Générique de la figure 5.5 après suppression et réinsertion de l'image b. Dans ce cas, la suppression et la réinsertion de b ne modifient pas l'Arbre Quaternaire Générique, car b était déjà fille de l'image a. En

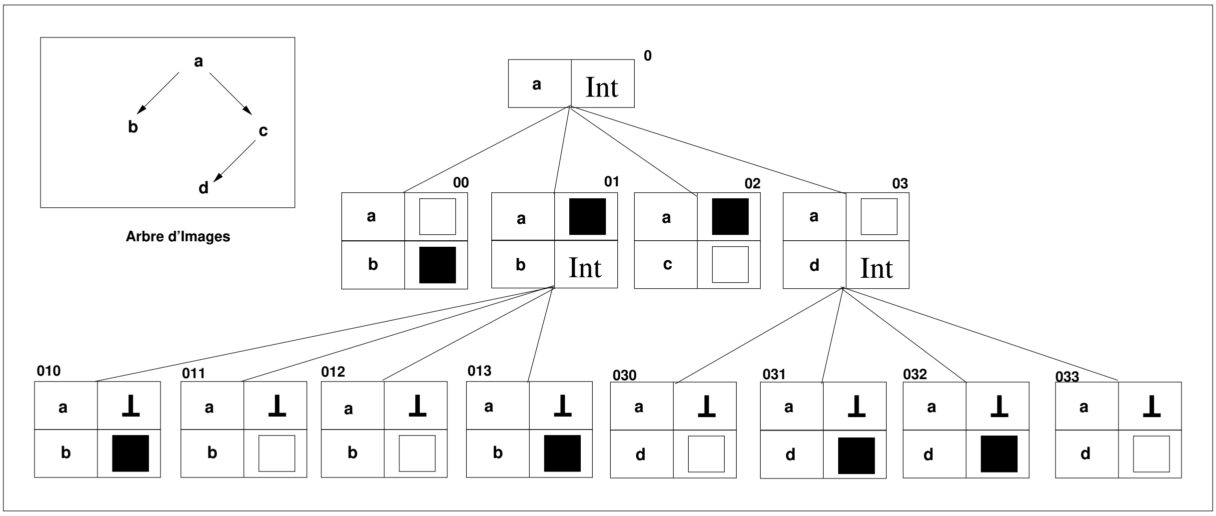


FIG. 5.11 – **La réorganisation de l'Arbre d'Images et de l'Arbre Quaternaire Générique par suppression et réinsertion de l'image** d.

revanche, elle a supprimé la liaison parentale entre b et d. Il semble plus intéressant, comme b est fille de la racine de l'Arbre d'Images, de déplacer l'image d et de laisser l'image b à son emplacement initial dans l'Arbre d'Images. Si l'image b avait eu plus d'images descendantes dans l'Arbre d'Images, les opérations de suppression et de réinsertion auraient permis de supprimer le lien parental, et donc le partage implicite, entre l'image b et ses descendants.

TAB. 5.3 – **Tableau de partage de l'image** d **avec les images** a, b, c

	a	b	c
$(0, Int)$	1	1	1
$(00, B)$	1	0	1
$(01, N)$	1	0	1
$(02, B)$	0	0	1
$(03, Int)$	0	0	0
$(030, B)$	0	0	0
$(031, N)$	0	0	0
$(032, B)$	0	0	0
$(033, N)$	0	0	0
Total	3	1	4

2. L'image d est supprimée. Comme d est feuille de l'Arbre d'Images, son déplacement dans l'Arbres d'Images et donc le calcul d'un nouvel identificateur interne est peu coûteux. En effet, il faut juste modifier l'identificateur interne de d dans les nœuds générique n où l'identificateur d apparaît. La figure 5.11 représente l'Arbre Quaternaire Générique après suppression et réinsertion de l'image d. L'image d est placée comme fille de l'image c avec laquelle la distance de Q-similarité est la plus proche de 0, comme le montre le tableau 5.3 (voir page

65)

5.2.6 Autres opérations

D'autres opérations comme le suivi de l'évolution d'une région dans différentes images, la comparaison, l'union, l'intersection ou la superposition de plusieurs images peuvent être appliquées aux images stockées dans un Arbre Quaternaire Générique.

a) Comparaison

La comparaison de deux images i et i' consiste à comparer les images région par région, c'est-à-dire à comparer les nœuds de même identificateur dans les arbres quaternaires des deux images. Pour tout nœud générique n, apparaissant dans l'arbre quaternaire de i et dans celui de i', la valeur v associée implicitement ou explicitement à i dans le nœud générique n est comparée à la valeur v' associée explicitement ou implicitement à i'. Si les images i et i' apparaissent dans la même ligne du nœud générique n ($v = v'$), le quadrant n a même valeur dans les deux images. En revanche, si les images i et i' apparaissent dans deux lignes différentes du nœud générique n ($v \neq v'$), le quadrant n a une valeur différente dans les deux images.

Dans une comparaison en détail des images, c'est-à-dire pour tous les nœuds de même identificateur apparaissant dans les arbres quaternaires des images. Plus précisément, pour chaque nœud générique n (en commençant par la racine, le nœud 0) :

1. Si i et i' sont associées à la même valeur v du nœud générique n, alors le quadrant n a même valeur dans les deux images, $v = v'$.

   (a) Si $v \neq Int$, le nœud n est feuille dans les arbres quaternaires de i et i'. L'algorithme reprend à l'étape 1 avec le nœud frère immédiatement suivant de n.

   (b) Si $v = Int$, le nœud n est interne dans les arbres quaternaires de i et i'. L'algorithme reprend avec le premier nœud fils de n.

2. Si i et i' sont associées respectivement aux valeurs v et v' du nœud générique n, $v \neq v'$, alors l'algorithme passe aux autres nœuds, la lecture de l'Arbre Quaternaire Générique se faisant en largeur d'abord.

Pour une comparaison moins complète, c'est-à-dire, sur quelques niveaux des arbres quaternaires, l'algorithme est sensiblement le même. L'utilisateur fixe le niveau p de profondeur jusqu'où les arbres quaternaires doivent être comparés et le système vérifie, à chaque étape, que le nœud n est situé à une profondeur inférieure à p dans l'Arbre Quaternaire Générique. L'utilisateur peut, en fonction du résultat obtenu, demander au système de poursuivre sa comparaison dans les niveaux inférieurs.

Exemple 5.11 La comparaison des images a et b dans l'Arbre Quaternaire de la figure 5.11 (voir page 65) est réalisée de la manière suivante :

1. Le nœud générique 0, racine de l'Arbre Quaternaire Générique est lu : l'image a est associé explicitement à la valeur Int, l'image b est associée implicitement à cette même valeur. Le nœud 0 est interne dans les arbres quaternaires des deux images *a* et *b*. L'algorithme se poursuit par conséquent avec le nœud 00.

2. L'image *a* est associée explicitement à la valeur *blanc* du nœud générique 00. En revanche, l'image *b* est associée explicitement à la valeur *noir*. Le quadrant 00 n'a pas la même valeur dans les deux images. L'algorithme se poursuit avec le nœud frère de 00, c'est-à-dire 01.

3. L'image *a* est associée explicitement à la valeur *noir* du nœud générique 01. En revanche, l'image *b* est associée explicitement à la valeur *Int*. Le quadrant 01 a donc une valeur différente dans les deux images.

4. L'image *a* est associée explicitement à la valeur *noir* du nœud générique 02. L'image *b* est associée implicitement à cette même valeur. Le quadrant 02 est noir dans les deux images. L'algorithme se poursuit avec le nœud frère de 02, c'est-à-dire 03.

5. L'image *a* est associée explicitement à la valeur *blanc* du nœud générique 03. L'image *b* est associée implicitement à cette même valeur. Le quadrant 03 est blanc dans les deux images. L'algorithme s'arrête.

Les images *a* et *b* ont donc les quadrants 02 et 03 en communs (ces quadrants ont même valeur dans les deux images). Si l'utilisateur avait demandé au système de comparer les nœuds uniquement jusqu'au niveau 1 de l'Arbre Quaternaire Générique, le résultat, dans ce cas particulier, aurait été le même.

Pour comparer la valeur d'une région dans différentes images, il suffit de lire le (ou les) nœud(s) générique(s) correspondant(s) aux plus petits quadrants incluant la région, et de voir quelles sont les valeurs de nœud d'arbre quaternaire de chaque image.

Pour comparer les images, il est également possible de calculer une similarité de surface des images. Cette similarité, appelée S-similarité, est calculable à partir des images stockées dans l'Arbre Quaternaire Générique. Elle est définie dans le chapitre 6 et est comparée aux similarités utilisées dans les approches de recherche d'images par le contenu.

b) Suivi d'une région dans plusieurs images

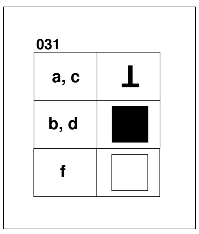


FIG. 5.12 – **Le suivi de la région 031 dans les images stockées dans l'Arbre Quaternaire Générique de la figure 5.9.**

Il est également possible, à la lecture d'un nœud générique, de suivre l'évolution de la région correspondante dans toutes les images de la base, c'est-à-dire de visualiser la valeur d'un quadrant dans plusieurs images. Une manière de visualiser ce type d'opération est d'afficher le nœud générique après avoir remplacé le partage implicite par du partage explicite, de sorte que tous les identificateurs d'images apparaissent clairement. Cette opération, nommée *explicitation d'un nœud*, est facilitée par la structure de l'Arbre Quaternaire Générique.

Exemple 5.12 Le suivi de la région 031 dans les images stockées dans l'Arbre Quaternaire Générique de la figure 5.9 (voir page 63) est représenté sur la figure 5.12. L'affichage du nœud générique 031 (après remplacement du partage implicite) permet de voir que le nœud 031 n'existe pas dans les arbres quaternaires des images a et c, qu'il est *noir* dans les arbres quaternaires des images b et d et *blanc* dans l'arbre quaternaire de l'image f. L'utilisateur, à la visualisation de la non existence de cette région dans les images a et c, peut demander au système de chercher la valeur de la région parente (c'est-à-dire 03) dans ces images.

Ce type d'opération est très utile dans les Systèmes d'Information Géographique où il est intéressant, par exemple, de comparer les images en fonction de leur temps de validité pour suivre l'évolution spatio-temporelle d'une région. En traitement d'images, le suivi de l'évolution d'une région permet d'analyser les conséquences d'une série d'opérations sur une région d'image (voir chapitre 3).

c) Construction d'images

Les opérations applicables aux images organisées en arbre quaternaire comme le calcul du complémentaire d'une image, l'union et l'intersection de deux images, ont été présentées dans le chapitre précédent (voir section 4.2.3). En utilisant les mécanismes de mise à jour, présentés en 5.2.2, il est facile d'appliquer ces opérations aux images stockées dans l'Arbre Quaternaire Générique. Ces opérations peuvent également être appliquées aux régions d'images. Il est possible de construire une image comme résultat du complémentaire d'une image. De même, il possible facile de construire une image résultant de l'union, de l'intersection, de la superposition ou de la différence de deux images.

La section 5.4, qui présente l'implantation des mécanismes de l'Arbre Quaternaire Générique, montre quelques copies d'écran de ce type d'opérations (voir pages 79-80).

5.3 Organisation d'un ensemble d'images en Arbre Quaternaire Générique

Les images représentées par un *Arbre Quaternaire Générique* sont organisées à l'aide d'une structure, appelée *Arbre d'Images*. L'opération d'insertion d'une image dans l'Arbre d'Images (voir section 5.2.3) permet de trouver l'image mère, c'est-à-dire l'image telle que la distance de Q-similarité soit la plus proche de 0 (voir section 5.1.2), d'une image créée ou modifiée à l'extérieur de la base, ou d'une image que l'on déplace dans l'Arbre d'Images.

Dans cette section, le problème est différent. Il s'agit de savoir comment organiser un ensemble d'images dont il n'existe pas, a priori, d'information sur leur Q-similarité potentielle. L'objectif est de proposer une solution [2] simple au problème d'organisation d'un ensemble d'images en Arbre Quaternaire Générique.

Cette section s'organise de la manière suivante. Les notations utilisées sont définies en 5.3.1. En 5.3.2, l'*Arbre Quaternaire Sans Partage Implicite* (SPI) est défini. En 5.3.3, la construction d'un graphe de partage entre images et la déduction de l'Arbre d'Images sont décrite. En 5.3.4, la règle permettant de construire l'Arbre Quaternaire Générique final est définie.

5.3.1 Notation

Dans la suite de la section, on utilise les ensembles suivants :

$\mathcal{I}_m$: l'ensemble des images à organiser en Arbre Quaternaire Générique

$\mathcal{N}$: l'ensemble des identificateurs de nœud (générique ou non)

$\mathcal{V}$: l'ensemble des valeurs de nœud. Pour les images en noir et blanc : $\mathcal{V} = \{0, 1, \mathtt{Int}\}$, avec 0 pour la couleur blanche, 1 pour la couleur noire et `Int` pour les nœuds internes.

On note $r \in \mathcal{I}_m$, l'image racine de l'Arbre d'Images.

De même, dans la suite de cette section, on fait appel aux fonctions suivantes :

$$\begin{array}{rcl} \mathcal{F} : \mathcal{I}_m \setminus \{r\} & \longrightarrow & \mathcal{I}_m \\ i & \longmapsto & i', \text{ l'image mère de } i \text{ dans l'Arbre d'Images} \end{array}$$

$$\begin{array}{rcl} \mathcal{V}_n : \mathcal{I}_m & \longrightarrow & \mathcal{V}, \\ i & \longmapsto & v, \text{ la valeur du nœud } n \text{ dans l'arbre quaternaire de } i \end{array}$$

Finalement, soit l'ensemble des lignes du nœud générique n :

$$\mathcal{L}_n = \{(\mathcal{I'}_m, v) \setminus \mathcal{I'}_m \subset \mathcal{I}_m,\ v \in \mathcal{V} \text{ et } \forall i \in \mathcal{I'}_m,\ \mathcal{V}_n(i) = v\}$$

Le formalisme de la règle de partage associée à l'Arbre d'Images (voir section 5.1.1) est le suivant :

Règle P : $\forall i \in \mathcal{I}_m \setminus \{r\}$, $\forall n \in \mathcal{N}$, si $\neg\ (\exists\ \mathcal{I'}_m \subset \mathcal{I}_m$ tel que $i \in \mathcal{I'}_m$et $(\mathcal{I'}_m, v) \in \mathcal{L}_n)$ alors $\mathcal{V}_n(i) = \mathcal{V}_n(\mathcal{F}(i))$

5.3.2 Arbre Quaternaire Générique SPI

L'*Arbre Quaternaire Générique SPI* (ou *Arbre Quaternaire Sans Partage Implicite*) est un Arbre Quaternaire Générique construit sans tenir compte du partage implicite éventuel entre les images, c'est-à-dire en considérant les images indépendamment les unes des autres. Cette structure est construite à partir des arbres quaternaires représentant les images.

2. Ce travail a été réalisé en collaboration avec Daniel Vanderpooten, Maître de Conférences à l'Université Paris-Dauphine.

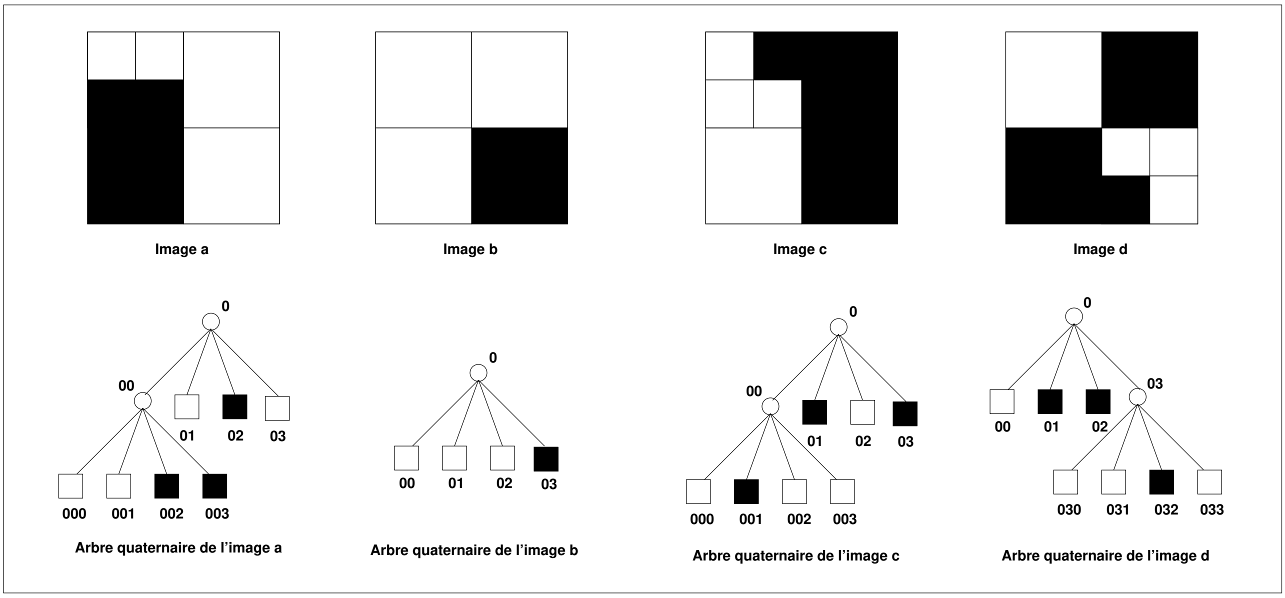


FIG. 5.13 – **La représentation de quatre images en arbre quaternaire.**

Chaque image de l'ensemble est représentée par un arbre quaternaire. Tous les nœuds des arbres quaternaires sont identifiés de la même manière (voir chapitre 4), de sorte que l'on puisse faire la correspondance entre les nœuds des différents arbres quaternaires. L'Arbre Quaternaire Générique SPI est construit avec autant de nœuds génériques qu'il y a de nœuds d'identificateurs différents dans l'ensemble des arbres quaternaires. Un nœud générique n a autant de lignes qu'un nœud n a de valeurs dans les arbres quaternaires. A chaque valeur v d'un nœud générique n est associée la liste des identificateurs d'image pour lesquelles l'arbre quaternaire correspondant contient un nœud identifié par n et de valeur v.

Exemple 5.13 La figure 5.13 représente quatre images organisées en arbre quaternaire. L'Arbre Quaternaire Générique SPI stockant les arbres quaternaires de ces images est représenté par la figure 5.14. Il a été construit sans tenir compte d'un ordre préalable entre les images, mais en tenant compte du partage explicite des valeurs de nœud entre les images.

Seul le nombre des identificateurs d'image associés à chaque valeur de nœud dépend de l'ordonnancement des images dans l'Arbre des Images. En effet, l'Arbre d'Images a pour objectif de diminuer le nombre d'identificateurs d'image apparaissant dans l'Arbre Quaternaire Générique, et donc dans l'Arbre Quaternaire Générique SPI. L'Arbre d'Images peut être déduit de l'Arbre Quaternaire Générique SPI en construisant un graphe de partage entre les images. La sous-section suivante présente les mécanismes de construction de ce graphe.

5.3.3 Graphe de partage et déduction de l'Arbre d'Images

a) Principes du graphe de partage

Un graphe complet non orienté $G = (\mathcal{I}_m, \mathcal{A})$ est construit à partir de l'Arbre Quaternaire Générique SPI. Ce graphe, nommé *graphe de partage*, est composé d'un ensemble $\mathcal{I}_m$ d'identificateurs

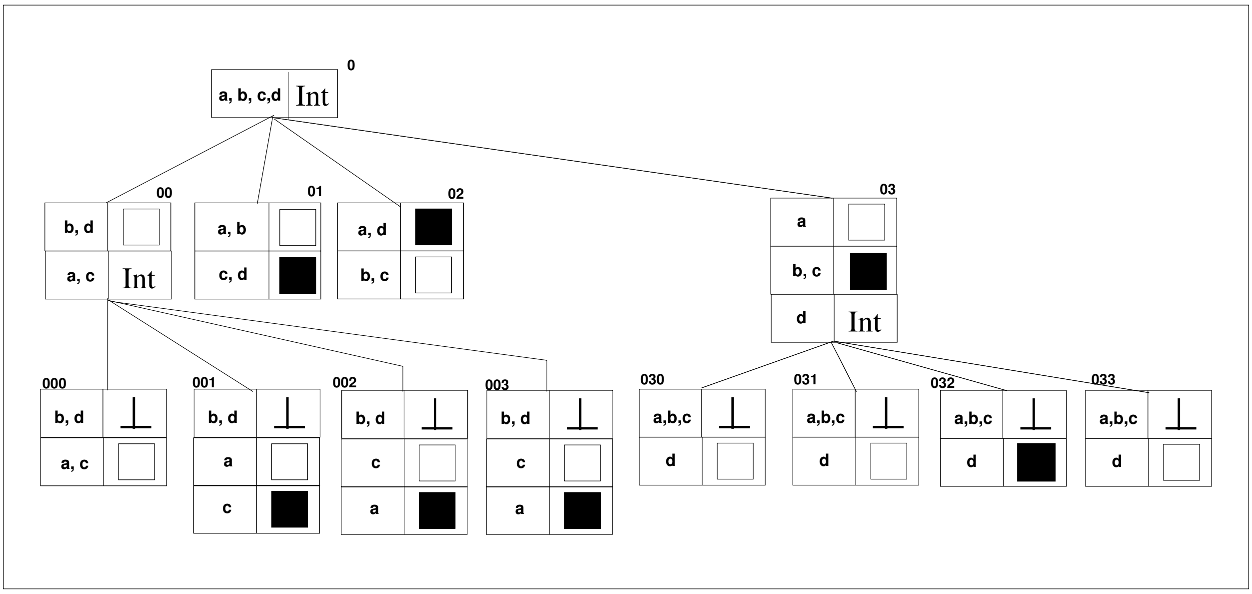


FIG. 5.14 – **L'Arbre Quaternaire Générique SPI des images de la figure 5.13.**

d'image et d'un ensemble $\mathcal{A}$ d'arêtes. Chaque arête $\{i, j\}$ est étiquetée par le nombre de nœuds communs (c'est-à-dire de même identificateur et de même valeur) dans les arbres quaternaires représentant les images i et j.

Le graphe G représente le partage explicite des valeurs de nœuds entre les images stockées dans l'Arbre Quaternaire SPI. Pour tout couple (i,i'), avec i et i' des identificateurs d'images stockées dans l'Arbre Quaternaire SPI $(i \neq i')$, le graphe de partage permet de savoir quel est le nombre de lignes de nœud générique où les identificateurs i et i' apparaissent simultanément. L'Arbre d'Images est déduit à partir du graphe G (voir sous-sections suivantes b) et c)).

Le graphe G est construit de la manière suivante. Il est composé d'autant de nœuds qu'il y a d'images dans l'ensemble à organiser. Dans un premier temps, les étiquettes de G sont toutes initialisées à zéro. Puis, elles sont augmentées au fur et à mesure de la lecture de l'Arbre Quaternaire Générique SPI. Pour cela, les identificateurs des images, apparaissant dans chaque ligne de nœud générique, sont ordonnés en un ensemble $\{i_1...i_m\}$, $m > 1$. Puis, pour chaque couple d'identificateurs d'image $(i_k, i_{k'})$, apparaissant dans une même ligne de nœud générique et tels que $k, k' \in [1, m]$ et $k > k'$, l'étiquette de l'arête $\{i_k, i_{k'}\}$ est augmentée de 1.

Exemple 5.14 Le graphe G, déduit de l'Arbre Quaternaire Générique SPI de la figure 5.14, est représenté en haut à gauche de la figure 5.15. En effet, les identificateurs des images a et b apparaissent simultanément dans une même ligne pour six nœuds génériques : 0, 01, $03x$, $x \in [0, 3]$. L'étiquette de l'arête $\{a, b\}$ est par conséquent 6.

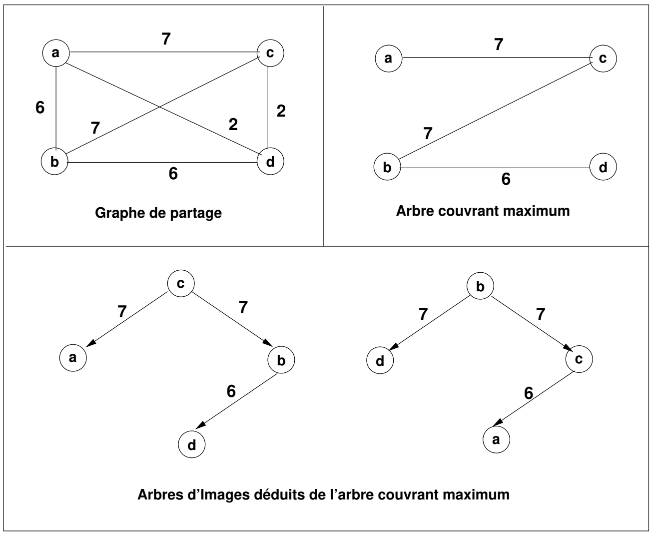


FIG. 5.15 – **Le graphe de partage, l'arbre couvrant maximum et les Arbres d'Images déduits de la figure 5.3.2.**

b) Recherche d'un arbre couvrant maximum

La recherche d'un arbre couvrant maximum du graphe G permet de connaître, pour chaque image i, avec quelle image de la base i partage le plus de valeur de nœuds. Un ou plusieurs arbres couvrants maximums peuvent être déduits du graphe G, en utilisant par exemple l'algorithme de Kruskal [Kru56]. L'algorithme de Kruskal consiste à ordonner les arêtes par ordre croissant des étiquettes et à éliminer les arêtes du graphe de plus petite étiquette tant que le graphe reste connexe [3].

Exemple 5.15 La recherche d'un arbre couvrant maximum du graphe G (voir figure 5.15 en haut à gauche), par application de l'algorithme de Kruskal est réalisé de la manière suivante. les arêtes du graphe G sont classées par valeur croissante des étiquettes : $\{a, d\}$, $\{c, d\}$, $\{a, b\}$, $\{b, d\}$,$\{a, c\}$, $\{b, c\}$. Tant que le graphe G reste connexe, les arêtes sont éliminées, en commençant par les arêtes de plus petite étiquette. Les arêtes $\{a, d\}$, $\{c, d\}$, $\{a, b\}$ peuvent être éliminées. L'arbre couvrant maximum final est représenté en haut à droite de la figure 5.15.

Si plusieurs arbres couvrants peuvent être déduits du graphe G, le choix d'un arbre couvrant plutôt qu'un autre ne change rien. Il est montré par la suite, dans la section 5.3.4, que la somme des étiquettes des arêtes de l'arbre couvrant maximum correspond au nombre d'identificateurs d'images pouvant être supprimés dans l'Arbre Quaternaire Générique SPI. Cette suppression d'identificateur correspond à transformation du partage explicite en partage implicite entre certaines images. Quel que soit l'arbre couvrant maximum choisi, la somme des arêtes a toujours la même valeur.

3. Un graphe est connexe s'il existe toujours une chaîne entre deux sommets quelconques du graphe. Une chaîne de longueur q est une séquence d'arêtes $(a_1, \ldots a_q)$ telle que : une extrémité de l'arête a_i coïncide avec celle de a_{i-1} et l'autre extrémité avec celle de a_{i+1}, $\forall i \in [2, q-1]$.

c) Déduction de l'Arbre d'Images

Une fois l'arbre couvrant choisi, on cherche un point de référence, c'est-à-dire une racine, afin d'en déduire un ou plusieurs Arbres d'Images possibles pour organiser l'ensemble d'images. Un point de référence est un point qui minimise la longueur des chemins dans l'arbre. Les approches de [KLCC77] permettent de déterminer un point de référence dans un arbre sans racine. L'arbre ainsi obtenu correspond à l'Arbre d'Images recherché pour organiser les images de la base.

Exemple 5.16 Les sommets b et c de l'arbre couvrant maximum du graphe G (voir figure 5.15 en haut à droite) peuvent être choisis comme point de référence, ou racine. Les deux Arbres d'Images possibles pour organiser les images a, b, c et d, sont représentés en bas de la figure 5.15.

Là encore, la manière de choisir un point de référence parmi plusieurs ne change rien au résultat final, c'est-à-dire au nombre d'identificateurs d'images supprimés dans l'Arbre Quaternaire Générique SPI. Le nombre d'identificateur supprimés correspond à la somme des étiquettes des arêtes de l'Arbre d'Images. Cette somme est la même quelle que soit la racine de l'arbre. De plus ample détails sont présentés dans la section 5.3.4.

Exemple 5.17 La figure 5.15 présente (en haut à droite) l'arbre couvrant maximum déduit du graphe de partage situé en haut à gauche de la figure. De cet arbre, il est possible de déduire deux racines : les nœuds b et c. Par conséquent, il y a deux Arbres d'Images possibles pour organiser les images a, b, c et d (voir en bas de la figure 5.15).

5.3.4 Construction de l'Arbre Quaternaire Générique final

On note P^{-1}, la règle qui permet, à partir de l'Arbre Quaternaire Générique SPI et d'un Arbre d'Images donné, de tenir compte du partage implicite entre les images. Cette règle permet de diminuer le nombre des identificateurs d'image dans l'Arbre Quaternaire SPI. Elle est notée P^{-1} en référence à la règle P, formalisée dans la section 5.3.1 et qui permet de retrouver le partage implicite entre images stockées dans un Arbre Quaternaire Générique non SPI.

La règle P^{-1} consiste à supprimer tous les identificateurs d'image apparaissant simultanément, dans une ligne de nœud générique, avec l'identificateur de leur image mère dans l'Arbre d'Images. Par conséquent, pour chaque nœud générique n, si une image i apparaît dans une même ligne de nœud générique que l'image j, i étant fille de j dans l'Arbre d'Images, alors il faut supprimer i de la ligne. Il est à noter que cette suppression doit se faire en lisant l'Arbre d'Images des feuilles vers la racine en largeur d'abord, c'est-à-dire des images filles vers leur mère, pour transformer au fur et à mesure le partage explicite entre mère et filles en partage implicite.

Le formalisme de la règle P^{-1} est donc le suivant :
$\forall n \in \mathcal{N} : \forall i \in \mathcal{I}_m \setminus \{r\}$, l'Arbre d'images étant lu, en largeur d'abord, des feuilles vers la racine, si $\mathcal{V}_n(i) = \mathcal{V}_n(\mathcal{F}(i))$, alors soit $\mathcal{I}'_m \subset \mathcal{I}_m$ tel que ($i \in \mathcal{I}'_m$et $(\mathcal{I}'_m, v) \in \mathcal{L}_n$), faire $\mathcal{I}'_m = \mathcal{I}'_m \setminus \{i\}$.

Il y a autant d'identificateurs d'image supprimés dans l'Arbre Quaternaire Générique que la somme des étiquettes des arêtes de l'arbre couvrant maximum choisi. En effet, le graphe de partage est étiqueté par le nombre de fois où deux identificateurs d'image apparaissent simultanément dans une même ligne de nœud générique. Un arc (i, j), dans l'Arbre d'Images, signifie que l'image j

est fille de l'image i. Une étiquette a_{ij} d'arc (i, j) signifie que les identificateurs des images i et j apparaissent simultanément dans a_{ij} lignes de nœud générique. Les images i et j peuvent donc partager implicitement la valeur des ces a_{ij} nœuds. Par conséquent, comme j est fille de l'image i dans l'Arbre d'Images, on peut supprimer les a_{ij} identificateurs j apparaissant simultanément avec i dans une ligne de nœud générique. Les identificateurs supprimés peuvent être retrouvés par application de la règle P sur l'Arbre Quaternaire Générique (voir section 5.3.1).

Comme il a été précisé dans la section précédente (5.3.3), le choix d'un arbre couvrant parmi plusieurs et le choix d'une racine de l'arbre parmi plusieurs ne changent rien au résultat final, c'est-à-dire au nombre d'identificateurs supprimés. Ces choix déterminent uniquement quels sont les identificateurs qui apparaissent dans les différents nœuds génériques ou plus précisément quels sont les identificateurs d'image qui vont être supprimés. Le nombre de ces identificateurs supprimés étant égale à la somme des étiquettes de l'Arbre d'Images.

Exemple 5.18 La figure 5.16 représente les deux Arbres Quaternaires Génériques (non SPI) permettant de stocker les images a, b, c et d. Ces Arbres Quaternaires Génériques ont été construits en appliquant la règle P^{-1} sur l'Arbre Quaternaire Générique SPI de la figure 5.14 et en se fixant un Arbre d'Images (situé à droite de la figure).

Lorsque l'image c est choisie comme racine de l'Arbre d'Images, l'application de la règle P^{-1} au nœud générique 0 de l'Arbre Quaternaire SPI de la figure 5.14 supprime les identificateurs a, b et d. En effet, pour chaque nœud générique, l'Arbre d'Images est lu des feuilles vers la racine et l'image d est la feuille la plus basse de l'Arbre. Or, $\mathcal{V}_0(d) = \mathcal{V}_0(\mathcal{F}(d)) = \mathcal{V}_0(b) = \mathcal{V}_0(\mathcal{F}(b)) = \mathcal{V}_0(c) = \texttt{Int}$, on peut donc supprimer les identificateurs d et b de $\mathcal{I'}_m$ avec $\mathcal{I'}_m = \{a, b, c, d\}$ et $(\mathcal{I'}_m, \texttt{Int}) \in \mathcal{L}_0$. Par conséquent, après une première application de la règle P^{-1} sur le nœud 0, $\mathcal{I'}_m = \{a, c\}$ avec $(\mathcal{I'}_m, \texttt{Int}) \in \mathcal{L}_0$. Au final, seul l'identificateur c est associé à la valeur Int du nœud générique 0 (voir figure 5.16).

Le nombre d'identificateurs supprimés dans l'Arbre Quaternaire Générique est égal à la somme des arêtes de l'arbre couvrant maximum, soit $7 + 7 + 6 = 20$. Ce nombre est indépendant du choix de la racine de l'Arbre d'Images.

5.4 Implantation des mécanismes

Cette section présente l'implantation des mécanismes de l'Arbre Quaternaire Générique. Un exemple d'organisation d'images médicales en Arbre Quaternaire Générique est présenté en 5.4.1. Le prototype, développé par des étudiants de l'Université Centrale du Venezuela en 1998 est présenté en 5.4.2. Ce prototype est décrit en détail dans [VR98, MC99, JMR+99b].

5.4.1 Images médicales organisées en Arbre Quaternaire Générique

L'Arbre Quaternaire Générique est appliqué dans ce chapitre à un exemple d'images en noir et blanc, dont la décomposition en arbre quaternaire se fait en fonction de l'homogénéité de la couleur. Cependant, le découpage des images en arbre quaternaire peut se faire selon plusieurs critères (voir section 4.2.2) : couleur, texture, etc. De même, le seuil et la manière de découper

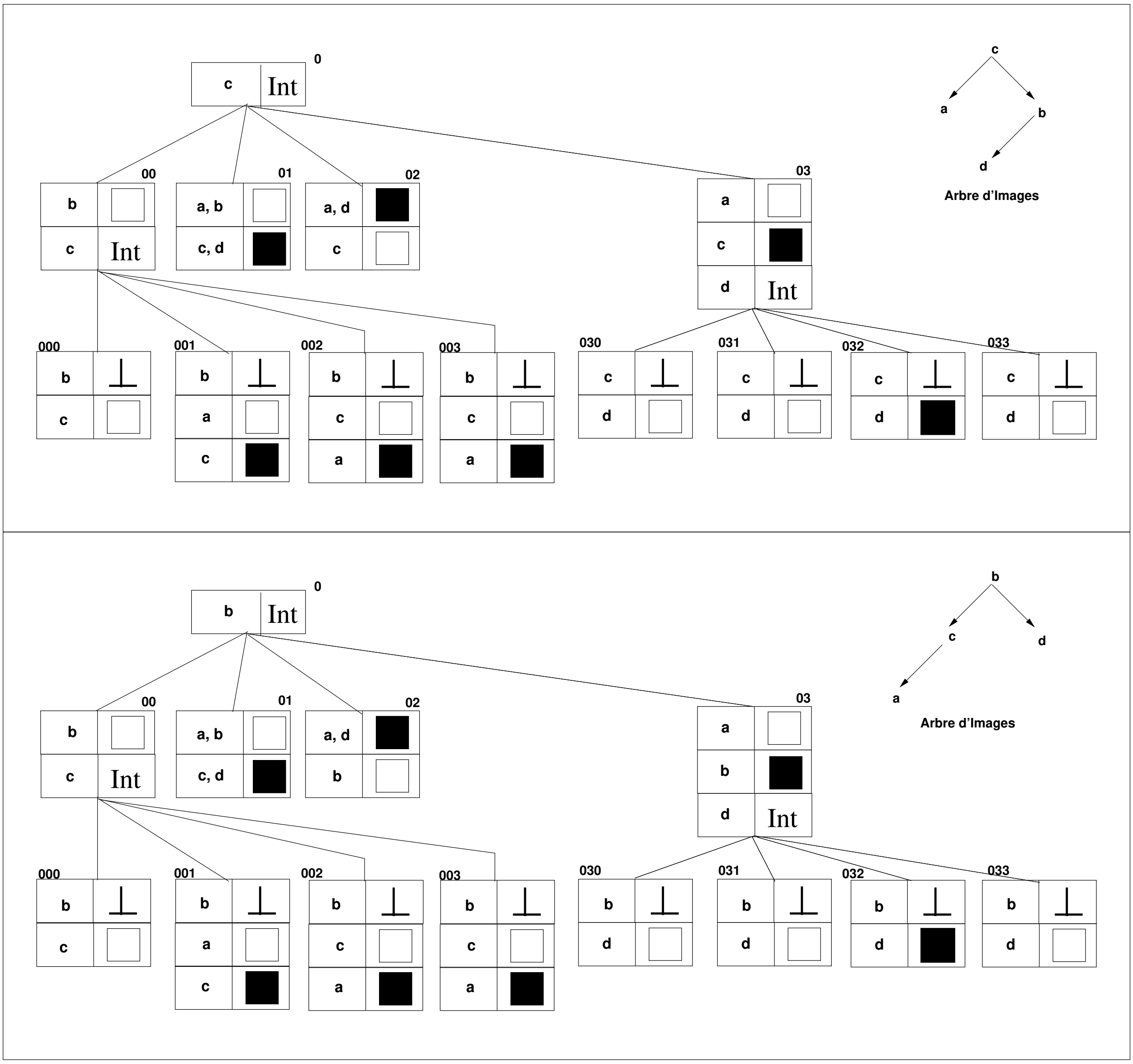


FIG. 5.16 – **L'organisation des images de la figure 5.13 en Arbre Quaternaire Générique.**

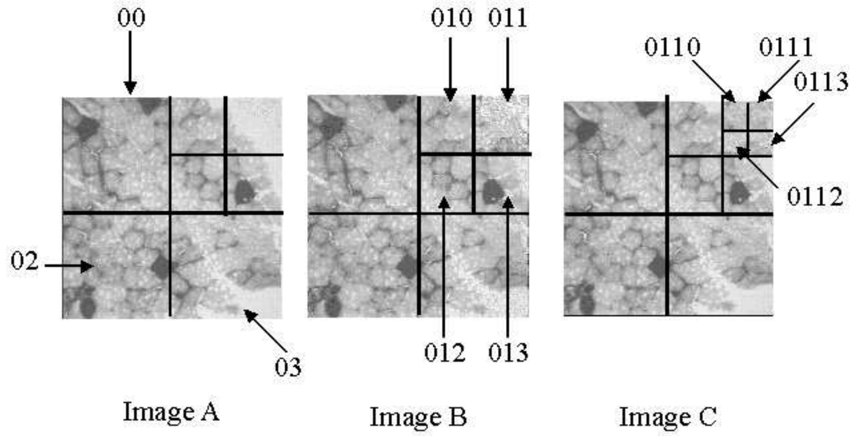


FIG. 5.17 – **Un exemple de trois images médicales en niveau de gris.**

chaque image en arbre quaternaire peuvent être choisis par l'utilisateur (voir figures 3.1 et 3.2). La figure 5.17 présente trois images médicales en niveaux de gris. La figure 5.18 représente ces trois images stockées dans un Arbre Quaternaire Générique.

5.4.2 Prototype MIS

Le prototype MIS (*sistema de Manipulación de Imàgenes Similares*) [JMR+99b] implante les mécanismes de l'Arbre Quaternaire Générique. Il s'agit d'un outil de traitement d'images dans lequel les utilisateurs choisissent le critère de découpage des images en arbre quaternaire et la manière d'organiser les images dans l'Arbre d'Images. Puisqu'il s'agit d'un atelier de traitement d'images, l'interaction entre le système MIS et l'utilisateur est très importante.

a) Description technique

Ce prototype [VR98] a été développé en langage de programmation C au dessus des librairies graphiques et de manipulation d'images *OPenGL* et *GLUT*. Les images gérées par le prototype sont des images Sun-Raster (*Bitmap*) en niveaux de gris. Toutes les images sont organisées en arbre quaternaire et stockées dans un Arbre Quaternaire Générique.

b) Interface

Le prototype se compose d'une fenêtre générale, découpée en deux parties, chaque partie pouvant accueillir une image (voir figure 5.19). L'utilisateur choisit les opérations qu'il souhaite appliquer à une image en sélectionnant l'image par un simple clic de souris et en choisissant un opération dans le menu, qu'il est possible d'obtenir par le bouton droit de la souris. Ce menu (voir figure 5.20) permet d'afficher une image, de sauvegarder une image sous forme d'image raster ou comme fille d'une image existante dans l'Arbre d'Images, de supprimer une image de la base, de découper une région en quatre ou de fusionner quatre quadrants d'images, et de réaliser des opérations (voir figure 5.19) sur l'image entière telles que le complémentaire, l'union, l'intersection,

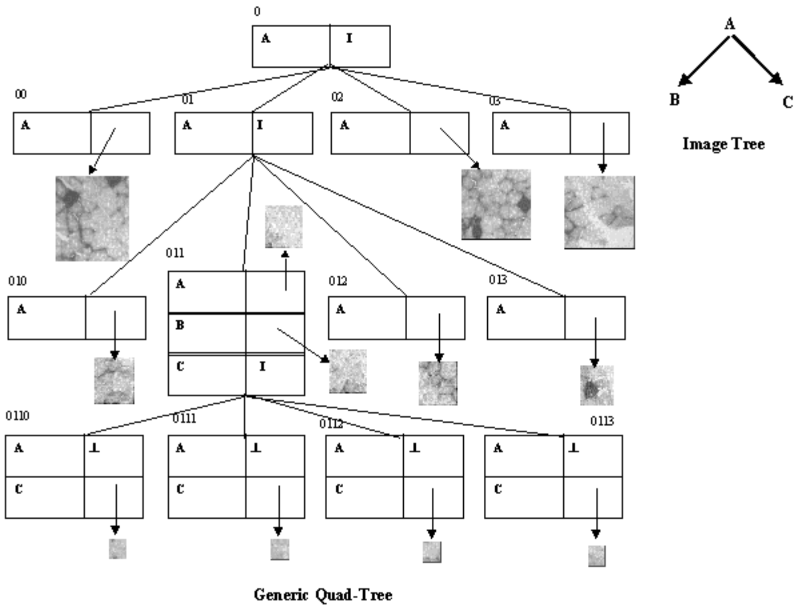


FIG. 5.18 – **Un Arbre Quaternaire Générique stockant les images de la figure 5.17.**

l'éclaircissement ou l'assombrissement de l'image, la dilatation etc. Un autre menu peut être obtenu en sélectionnant une région à partir du bouton gauche de la souris. Ce menu permet de faire des opérations sur la région sélectionnée, c'est-à-dire de calculer le complémentaire de la région, la diviser en quatre, etc.

Le suivi d'une région d'image, et l'organisation automatique de l'Arbre d'Images (sans l'intervention de l'utilisateur) n'ont pas encore été implantés.

c) Exemple d'opérations réalisables sous MIS

La figure 5.21 présente le résultat de l'application de l'opération complémentaire sur une région d'image. L'image originale d'un babouin est située à droite de la copie d'écran. L'image située à gauche correspond à l'image initiale sur laquelle l'opération complémentaire a été appliquée. Cette opération n'a pas été appliquée sur l'image entière, mais uniquement sur la région 01 de l'image, correspondant à l'œil droit du babouin.

La figure 5.22 présente (à droite), le résultat de l'union de l'image du babouin avec une image de cellules (située à gauche de la figure).

Conclusion

Le stockage des images est un problème. En effet, même si les mémoires ou les disques ont une taille de plus en plus grande, les images occupent beaucoup de place et les applications, comme l'imagerie médicale, génèrent un grand nombre d'images. Un département typique de radiologie

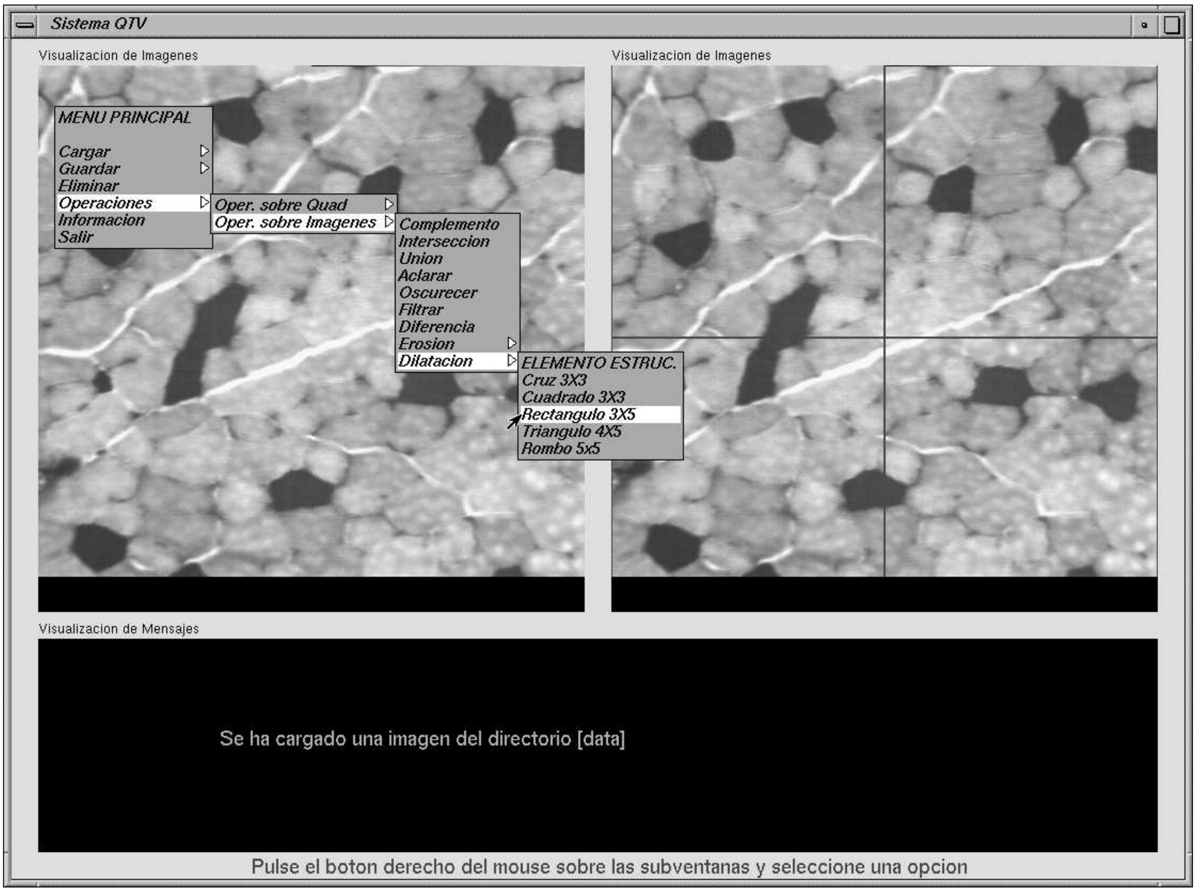


FIG. 5.19 – **L'écran général du prototype MIS.**

par exemple génère par an, selon [KSF+96], entre 100000 et 10000000 images ; le centre médical de Baltimore génère annuellement 1,5 téta-octets d'images.

L'Arbre Quaternaire Générique, présenté dans ce chapitre, permet d'optimiser le stockage d'images similaires organisées en arbre quaternaire, en évitant de stocker plusieurs fois des morceaux communs à plusieurs images. Cette optimisation, réalisée par le mécanisme de partage explicite, est d'autant plus considérable que les morceaux partagés ont une grande taille (dans le cas d'images en niveau de gris par exemple). Le partage implicite permet, quand à lui, de diminuer le nombre d'identificateurs d'image apparaissant dans les nœuds génériques. Il diminue donc la taille des nœuds génériques.

Dans le cas d'images en niveaux de gris, le partage implicite ne va pas faire gagner énormément de place par rapport au partage explicite. Si le critère de découpage des images est la couleur, et si seule la couleur est stockée au niveau des nœuds génériques, dans ce cas, le partage implicite permet d'éliminer beaucoup d'identificateurs d'image dans l'Arbre Quaternaire Générique, car le partage de valeurs est plus important que lorsque des morceaux d'images sont stockés. La quantité d'information partagée dépend donc de la manière avec laquelle sont découpées les images et

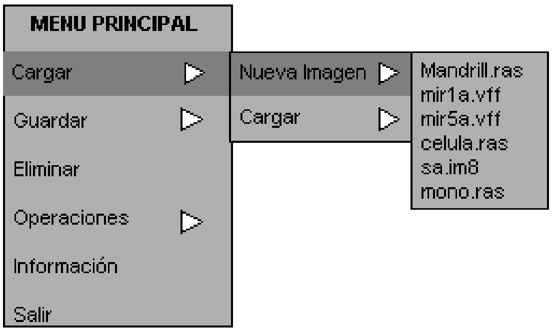


FIG. 5.20 – **Le menu général du prototype MIS.**

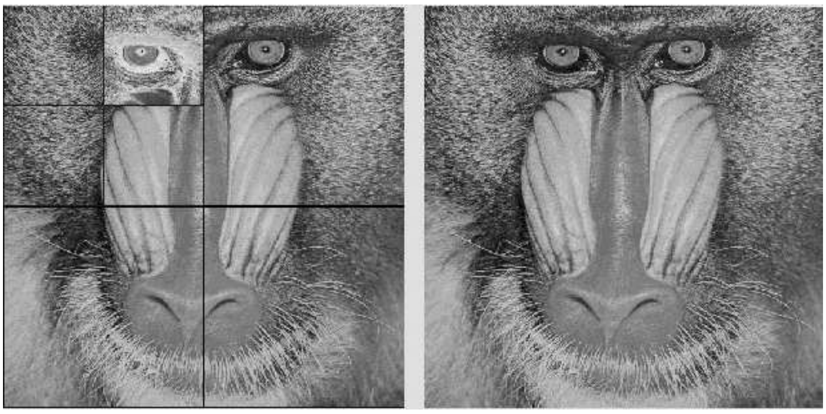


FIG. 5.21 – **Le complémentaire d'une région d'image.**

de l'information stockée au niveau des nœuds d'arbre quaternaire. Dans tous les cas, le partage explicite de valeurs permet d'optimiser le stockage des valeurs et le partage implicite permet d'optimiser le stockage d'identificateurs de nœud.

Dans ce chapitre, la valeur des nœuds interne est *Int*. Cependant, il peut être intéressant, pour comparer les images et accélérer leur lecture (permettre une lecture par niveau de précision, par exemple), de stocker de l'information sur les nœuds fils du nœud, comme un motif, ou la moyenne des valeurs de ces fils par exemple.

Les mécanismes proposés dans ce chapitre correspondent à un codage d'un ensemble d'images. Ils se différencient des mécanismes et des techniques de compactage ou de compression d'images comme GIF [Com87] ou JPEG [Wal89, Wal91], car ces techniques ne sont associées qu'à une seule image. Les standards MPEG (*Moving Pictures Expert Group*) [LG91] ou GIF animés s'adressent, en revanche, à plusieurs images mais ces images sont organisées en séquence. Dans le cas de l'Arbre Quaternaire Générique, les images ne sont pas organisées en séquence mais en arborescence. Toutefois, il est possible d'inclure des techniques de compression à l'Arbre Quaternaire Générique, en les appliquant par exemple aux morceaux d'images stockés dans l'arbre.

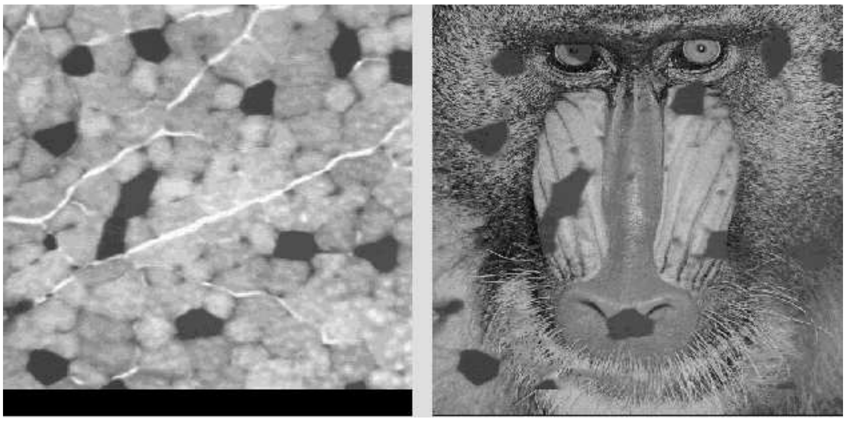


FIG. 5.22 – **L'union de deux images.**

L'autre objectif de l'Arbre Quaternaire Générique est de permettre d'appliquer des opérations aux images. Cette structure permet de lire une image, d'insérer ou de supprimer une image dans l'Arbre Quaternaire Générique, de modifier une image ou plusieurs images simultanément, de comparer des images et de faire le suivi de l'évolution de région d'images. Ces types d'opération sont très importants dans le domaine des ateliers de traitement d'images (voir chapitre 3).

Dans l'approche proposée, les opérations sont appliquées sur des images organisées en arbre quaternaire. La force de l'arbre quaternaire est qu'il offre un repère absolu des régions d'image et qu'il permet de paralléliser certaines opérations. Après avoir re-cadrer les images et les avoir placées dans l'Arbre Quaternaire Générique, il est possible de faire du suivi d'évolution de mêmes zones. Ce suivi (voir section 5.2.6) est intéressant dans le domaine du traitements d'images où il est possible de visualiser les conséquences de différents traitements sur une même régions. De même, ce type d'opération est important dans le domaine spatio-temporel, pour étudier l'évolution d'une zone géographique (par exemple le mitage des constructions).

Même si les mécanismes proposés ont été jusqu'à aujourd'hui appliqués à des images médicales, dans le cadre d'un atelier de traitement d'images, il est clair que l'Arbre Quaternaire Générique peut avoir un atout pour d'autres types d'images. Dans le domaine géographique, par exemple, l'Arbre Quaternaire Générique peut faciliter la gestion de fonds de carte sur lesquels travaillent les utilisateurs de Systèmes d'Information Géographiques. En outre, l'arbre quaternaire est très utilisé dans le domaine de la recherche d'images par le contenu [SC94, LOT94, RSZ96, SGZR97, MBNW99]. Par conséquent, il est envisageable d'inclure l'Arbre Quaternaire Générique ou les mécanismes qui en découlent à un système de recherche d'images par le contenu.

Le chapitre qui suit compare les mécanismes de l'Arbre Quaternaire Générique avec les travaux de la recherche en présentant notamment un mesure de similarité de surface d'images organisées en Arbre Quaternaire Générique.

Chapitre 6

Comparaison avec les autres travaux de recherche

Résumé

Il existe, dans le domaine des images, deux grandes catégories d'approches : celles qui recherchent des images dans une base d'images en fonction de l'information disponible sur leur contenu, et celles qui optimisent le stockage des images. Les approches de chevauchement d'arbres quaternaires font partie de la deuxième catégorie. Ce chapitre compare l'Arbre Quaternaire Générique à ces deux catégories d'approches, en présentant en particulier un état de l'art sur les approches de recherche d'images par le contenu ou de recherche par similarité. Pour placer l'Arbre Quaternaire Générique par rapport aux approches de recherche par similarité d'images, une nouvelle similarité est définie, la *S-similarité*. Cette similarité correspond à une similarité visuelle des images stockées dans un Arbre Quaternaire Générique. Ce chapitre montre que l'Arbre Quaternaire Générique peut s'intégrer dans un système de recherche d'images par le contenu. Il compare également l'Arbre Quaternaire Générique avec les approches de chevauchement dont il est très proche, du point de vue optimisation du stockage, mais très éloigné du point de vue des opérations applicables aux images.

Introduction

Traditionnellement, des attributs textuels ou des méta-données de l'image tels que le nom du fichier, la légende ou des mots-clés ont été utilisés pour indexer les images, c'est-à-dire sélectionner, dans une collection d'images, les images les plus pertinentes pour l'utilisateur [AY99]. Mais ce type de représentation nécessite l'intervention manuelle d'un opérateur et la sémantique des mots-clés choisis est parfois inadéquate, approximative et subjective [Nas97].

Depuis le début de la décennie, les chercheurs se posent le problème de l'indexation automatique des images fondée sur leurs caractéristiques visuelles et non-visuelles afin de permettre la *recherche d'images par le contenu* (en anglais *content-based image retrieval*). Il s'agit de requêtes sur la *similarité* des images, appelée également requêtes de voisinage (*nearest neighbor*).

Ces approches ne tiennent pas compte de l'espace occupé par les images. En revanche, les approches de chevauchement, présentées dans le chapitre 4 permettent d'optimiser le stockage d'images similaires. L'Arbre Quaternaire Générique est comparé à ces différentes approches.

L'organisation de ce chapitre est la suivante. La section 6.1 présente un rapide état de l'art sur la recherche d'images similaires. La similarité visuelle des images organisées en Arbre Quaternaire Générique est définie dans la section 6.2. Finalement, l'Arbre Quaternaire est comparé avec les autres approches dans la section 6.3.

6.1 Similarité dans le domaine des images : un état de l'art

Il existe plusieurs manières de rechercher, dans une base d'images, une image similaire à une image requête. La définition de la similarité dépend beaucoup de la manière avec laquelle l'image est recherchée. Il n'existe pas, à notre connaissance, de définition générale de la similarité entre images. La similarité est d'autant plus difficile à définir qu'elle dépend énormément des utilisateurs qui n'ont pas tous le même point de vue sur ce *qu'est une image similaire* [ABH97, LHD97].

Cette section présente un état de l'art sur la recherche d'images par la contenu. La sous-section 6.1.1 présente une classification des méthodes d'interrogation des images. La sous-section 6.1.2 décrit plusieurs approches utilisant des index multidimensionnels pour accélérer la recherche d'images. La sous-section 6.1.3 présente quelques approches permettant de faire des requêtes sur la similarité des régions d'image plutôt que la similarité des images en entier.

6.1.1 Classification des méthodes d'interrogation d'images

Les requêtes de similarité sur les images peuvent être classées de la manière suivante [AY99] :

- **Requête sur une seule caractéristique** : L'utilisateur précise les pourcentages de différentes valeurs prises par la caractéristique. Il s'agit de requêtes de la forme "Retrouver toutes les images contenant 25% de rouge, 50% de vert et 25% de bleu.". Ce type de requêtes est proposé, par exemple, dans le système QBIC [FSN+95].

- **Requête simultanée sur plusieurs caractéristiques** : L'utilisateur combine différentes caractéristiques. Il s'agit par exemple de requêtes telles que "Retrouver les images contenant 75% de la couleur verte et 25% de la texture d'arbre.". Dans ce type de requête, comme dans le précédent, l'utilisateur peut choisir les caractéristiques (couleur, texture, forme) auxquelles doivent répondre les images résultats, dans la palette qui lui est offerte. Le système Virage [Vir] propose de combiner plusieurs caractéristiques pour déterminer la similarité des images.

- **Requête sur la localisation des caractéristiques** : L'utilisateur spécifie l'emplacement et les valeurs des caractéristiques dans l'image. Par exemple, "Retrouver les images dont la couleur bleue est situé en haut et la couleur verte est située en bas à droite de l'image". Ce type de requêtes est proposé, par exemple, dans le système QBIC [FSN+95].

- **Requête par l'exemple** *(query by example)* : L'utilisateur sélectionne une image dans la base et souhaite retrouver toutes les images de la base similaires à l'image extraite. La similarité est définie par rapport aux caractéristiques de l'image, ces caractéristiques pouvant être choisies par l'utilisateur. Ce type de requête est de la forme "Retrouver les images dont la texture est similaire à celle de telle image.". Le système Surfimage [NBMM98], par exemple, propose ce type de requêtes.

- **Recherche d'image par croquis** *(query by sketch)* : L'utilisateur schématise grossièrement le contenu de l'image, en utilisant par exemple des objets conceptuels [ABH97]. La classe de ces objets ainsi que leur position relative est prise en compte pour rechercher les images similaires, c'est-à-dire contenant les mêmes objets aux mêmes positions. Le système VisualSeek [SC96] permet de faire des recherches d'images par croquis.

- **Recherche d'image par objets** : L'utilisateur peut décrire les caractéristiques d'un objet plutôt que celle de l'image entière. Il s'agit par exemple de "Retrouver les images contenant une voiture rouge". Il est possible de combiner la recherche de plusieurs objets et de spécifier les relations spatiales entre les différents objets. On obtient des requêtes de la forme "Retrouver les images où Clinton est assis à coté de Chirac.". Le système DISIMA [OTOL+97], par exemple, permet d'interroger les images sur les objets qu'elles contiennent.

- **Requête sur les méta-données de l'image** : L'utilisateur interroge les images en spécifiant les valeurs des attributs textuels décrivant l'image, comme par exemple "Retrouver les images de résolution 300 dpi (*dots per inch*) créées en 1985.".

Pour accélérer les recherches, des structures multidimensionnelles comme les arbres Rectangle (ou *R-tree*) [PF95, PF97] ou les arbres quaternaires [SC94, MBNW99] sont souvent utilisées. La section suivante présente plusieurs approches utilisant ce type de structures.

6.1.2 Recherche d'images par une structure multidimensionnelle

Les structures d'index ou les structures multidimensionnelles peuvent être utilisées pour faciliter les recherches d'images similaires. Il existe, à notre connaissance, deux grandes familles d'approches utilisant ce type de structures. La première catégorie correspond à des approches utilisant les structures multidimensionnelles pour accélérer les recherches de similarité entre images, ces recherches étant basées sur un nombre limité de caractéristiques. L'autre catégorie correspond aux approches qui représentent les images par un point ou un vecteur dans un espace à plusieurs dimensions et qui déplacent le problème de la recherche d'images similaires vers un problème de recherche de voisinage dans un espace multidimensionnel.

Cette section présente succinctement ces deux catégories d'approches.

a) Accélération de la recherche d'images par une structure d'index

Comme il a été précisé dans la section 4.2.2, plusieurs approches utilisent des structures comme les arbres quaternaires pour extraire de l'information des images telle que la texture [SC94, SZB99]. Certaines approches [RSZ96, SGZR97] utilisent, à cet effet, des adaptations de l'arbre quaternaire basées sur une décomposition récursive de l'image en cinq ou huit régions.

Les auteurs de [LOT94] proposent d'utiliser l'arbre quaternaire pour aider à la recherche d'images similaires. L'arbre quaternaire est utilisé dans [LOT94] pour positionner les couleurs de l'image. Un histogramme de couleurs de l'image entière est associé à la racine de l'arbre quaternaire. Puis, à chaque nœud est associé l'histogramme de la région correspondante dans l'image. Cette structure d'arbre quaternaire est appelé *multi-level color histogram*. La recherche de similarité entre images se fait par niveau, à l'aide d'une fonction de similarité (ou distance) sur les histogrammes. La comparaison de deux images se fait tout d'abord en comparant les histogrammes associés aux racines des arbres quaternaires des deux images, c'est-à-dire en comparant les histogrammes des deux images entières. Puis, si la similarité est correcte (par rapport à un certain seuil), la similarité des histogrammes est récursivement calculée pour les histogrammes situés à un niveau en dessous dans l'arbre quaternaire. Cette technique permet d'éliminer, à chaque niveau, des images considérées comme non similaires (du point de vue des histogrammes de couleurs). Il s'agit, par conséquent, d'un filtrage des images. Un arbre R est utilisé dans cette approche pour organiser les images en classes d'images ayant les mêmes couleurs dominantes, afin d'élaguer au maximum le nombre d'images à comparer. Un dernier index, un arbre B, est utilisé pour indexer les attributs textuels associés aux images. Les trois index (arbre B, arbre R et arbre quaternaire ou *multi-level color histogram*) peuvent être combinés pour former un index à trois niveaux d'abstraction (*3-Tier Approach*).

Certaines approches [PF95, PF97, NTOL97, OTOL$^+$97, NTOL99] utilisent les index pour accélérer les recherches sur la similarité des positions spatiales des objets composant les images. Le système DISIMA (*DIStributed Image database MAnagement system*) [OTOL$^+$97], par exemple, représente le contenu d'une image par un ensemble d'objets saillants (*salient objects*), c'est-à-dire d'objets ayant un sens pour l'utilisateur ou l'application. Un structure, nommée $2D - string$, est utilisée pour représenter, sous forme de chaînes de caractères, les relations spatiales entre les objets d'une image. Par exemple, l'image de la figure 6.1 est représentée par la structure $2D - string$ suivante : ($arbre < (maison : (chien = soleil))$; $(maison : chien) < arbre < soleil$). Cette structure permet de spécifier le contenu de l'image. Les objets saillants sont identifiés par des noms de concepts (*arbre*, *maison*, etc.). Les symboles $<$, : et $=$ permettent de spécifier respectivement, si un objet est situé à droite, à gauche, en dessous ou au dessus d'un autre objet, si des objets font partie d'un même ensemble et s'ils ont la même localisation spatiale. La première chaîne, $arbre < (maison : (chien = soleil))$, correspond aux localisations verticales des objets : l'*arbre*

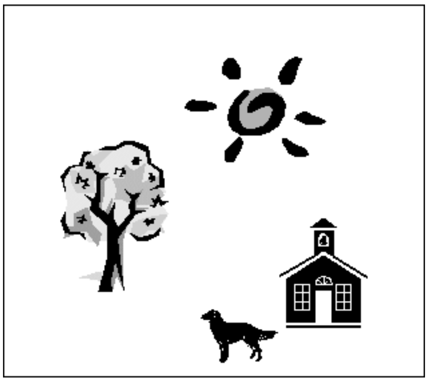


FIG. 6.1 – **Une image.**

est situé à gauche de l'ensemble formé par la *maison*, le *chien* et le *soleil*, le *chien* et le *soleil* ayant la même localisation verticale. La deuxième chaîne, $(maison : chien) < arbre < soleil$, correspond aux localisations horizontales des objets : la *maison* et le *chien* sont situés en dessous de l'*arbre*, lui-même situé en dessous du *soleil*. Pour organiser les $2D - string$ et permettre leur comparaison, les auteurs de [NTOL97] proposent une structure hiérarchique, le $2-D-S-tree$. En outre, les objets saillants sont organisés en hiérarchie de concepts permettant de classer les images selon une clé d'index, qui peut être, par exemple, la couleur. Si la requête consiste à "Trouver les images où une voiture est située à droite d'un chien noir" [NTOL97], dans ce cas, la hiérarchie de concepts permet d'éliminer un certain nombre d'images en sélectionnant celles qui contiennent un chien noir. La structure $2 - D - S - tree$ permet de trouver les images contenant une voiture à droite d'un chien. Le résultat de la requête se fait par intersection des deux ensembles d'images trouvés.

Dans le même ordre d'idée, le contenu des images est représenté dans [PF95, PF97] par des graphes ARG (ou *Attributed Relational Graphs*). Chaque nœud du graphe décrit un objet (sa taille, son orientation etc.) et les arcs du graphe représentent les relations spatiales entre les différents objets de l'image, c'est-à-dire la distance qui les sépare et leur position relative. La requête de similarité sur les images devient par conséquent une requête de similarité sur les graphes.

Les auteurs de [PF95, PF97] proposent également de représenter chaque graphe ARG par un point à plusieurs dimensions et d'utiliser un arbre R pour indexer ces points. La sous-section suivante présente plusieurs approches proposant des mécanismes similaires.

b) Requêtes de voisinage

L'idée générale des approches des requêtes de voisinage consiste à extraire k caractéristiques d'une image et à représenter ces caractéristiques par un vecteur ou un point à k dimensions [KSF+96]. Les points sont indexés par des méthodes d'indexation multidimensionnelles comme les structures d'accès spatial ou SAM (*Spatial Access Method*).

La requête "Trouver les images similaires à une image i" devient par conséquent une requête de la forme "Trouver les points de l'espace qui sont proches du point i". Il s'agit d'une requête d'intervalles (ou *range query*, "Trouver les points situés à une distance ε de i") ou d'une requête de voisinage (*nearest neighbor query*, "Trouver les v plus proches voisins de i") [KSF+96]. Le problème est équivalent aux recherches de séquences temporelles similaires [Fal99].

De nombreuses approches ont été proposées pour répondre aux problèmes de recherche de voisins dans un espace multidimensionnel. Plusieurs approches [RKV95, CF98] utilisent des index spatiaux, comme les arbres R, pour organiser les vecteurs multidimensionnels et permettre de les interroger. Les auteurs de [RKV95] proposent une approche de requêtes de voisinage basée sur les arbres R*. Un arbre R* [BKSS90] diffère principalement d'un arbre R classique par l'algorithme d'insertion [Gün98, GG98]. La différence se situe au niveau de l'éclatement des nœuds. Les auteurs de [WJ96] proposent un index dit de similarité qui utilise des sphères englobantes minimum plutôt que des rectangles comme dans les arbres R.

Les approches basées sur les index spatiaux appliquent en général des méthodes de partitionnement de l'espace de sorte que chaque partition contienne approximativement le même nombre de points et qu'une requête de voisinage ne touche qu'un petit nombre de partitions [Chi94]. Il est possible d'utiliser des arbres tels que le $VP-tree$, qui est basé sur le partitionnement de l'espace en fonction des distances relatives des points par rapport à un point particulier donné (*vantage point*) [Chi94].

Toutefois, certains auteurs [BO97, CPZ97] pensent que les structures spatiales ne sont pas idéales pour gérer les points multidimensionnels. Selon [BO97], l'arbre R n'est plus efficace au delà de 20 dimensions. Les auteurs de [BKK96] appliquent, par conséquent, l'algorithme de [RKV95] sur une structure plus spécialisée dans le domaine multidimensionnel, l'arbre X.

Le problème lors de l'utilisation des index spatiaux est qu'il n'y a pas de corrélation entre les caractéristiques représentées par un vecteur [CPZ97]. Certaines auteurs [CPZ97] considèrent qu'il est plus judicieux de prendre en compte la distance relative entre les points multidimensionnels, en utilisant un arbre métrique tel que l'arbre M, plutôt que de partitionner l'espace. Les approches basées sur les arbres métriques construisent une hiérarchie de groupes de points, c'est-à-dire un arbre de sphères, chaque sphère incluant des sous-sphères, etc. [Fal99].

La majorité des approches présentées précédemment utilise une signature (voir section d)) ou une combinaison de signatures [NBMM98] pour chaque image ; il peut s'agir de signatures sur la couleur [LOT94], sur la texture [SC94] ou de signatures plus complexes basées sur les ondelettes qui permettent de capturer à la fois la couleur, la texture et la forme d'une image [NRS99]. Ces approches calculent la similarité des images en considérant les images entières. La sous-section qui suit présente quelques approches proposant la recherche d'images par régions similaires.

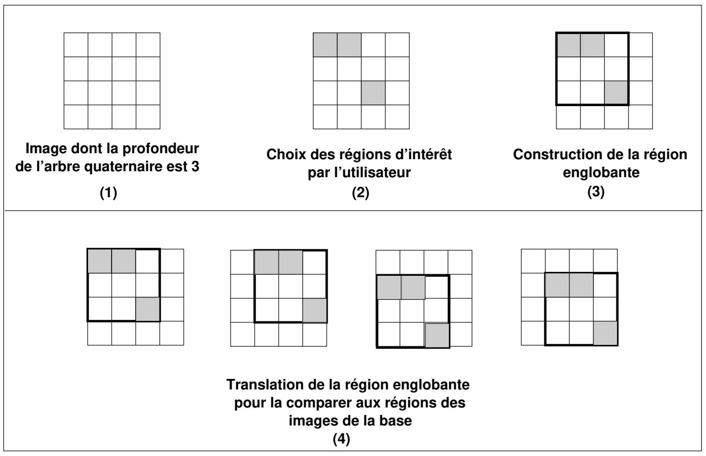


Fig. 6.2 – **Les étapes de la recherche d'images par région dans [MBNW99].**

6.1.3 Requêtes sur les régions d'images

Il existe, à notre connaissance, peu d'approches [Smi97, MBNW99, NRS99] permettant de calculer la similarité de deux images à partir de la similarité de leurs régions. Traditionnellement la recherche des images similaires à une image requête retourne les images dont la signature est la plus proche de celle de l'image requête. Le problème est que, lorsque seules quelques régions sont similaires dans les images, un tel type de recherche ne fonctionne pas [NRS99]. L'idée est par conséquent de découper l'image en régions et de calculer une similarité des images basée sur la similarité des régions.

A cet effet, plusieurs mécanismes de découpage d'une image en région ont été proposés. Les auteurs de [MBNW99] considèrent que l'arbre quaternaire est une structure appropriée pour diviser une image en sous-régions. Dans leur approche, chaque image est découpée en N sous-images en utilisant un arbre quaternaire dont le niveau de profondeur est choisi par l'utilisateur (voir étape (1) sur la figure 6.2). Dans l'approche proposée par [Smi97], les images sont découpées en régions à partir d'une librairie de motifs préfixés. Le découpage des images se fait en fonction de la texture et de la couleur des motifs. Dans WALRUS (*WAveLet-based Retrieval of User-specified Scenes*) de [NRS99], les images sont découpées à partir de fenêtres dont la taille est fixée par l'utilisateur.

Dans l'approche de [MBNW99], les caractéristiques de chaque sous-image ou région sont combinées et représentées par un vecteur multidimensionnel. L'utilisateur choisit des régions d'intérêt dans une image requête (voir étape (2) sur la figure 6.2). Le système construit une région minimum englobante des régions sélectionnées par l'utilisateur (voir étape (3) sur la figure 6.2), afin de déterminer quelles sont les plus petites régions pouvant contenir les régions choisies pour la requête. Puis, les images de la base sont comparées avec l'image requête en offrant la possibilité d'effecter des translations sur la région englobante dans l'image requête (voir étape (4) sur la figure 6.2). En effet, l'arbre quaternaire permet d'effectuer des translations de morceaux d'image par déplacement de sous-arbres [Sam84].

L'approche de [NRS99] offre à l'utilisateur, en plus du choix de la taille de la fenêtre de découpage de l'image, la possibilité d'initialiser deux paramètres : ε et ξ. Deux régions sont considérées comme similaires, si la distance de leur signature (basée sur les ondelettes) est inférieure à ε. Deux images sont considérées comme similaires si elles ont un ensemble de régions similaires et si la fraction de la surface représentée par ces régions par rapport à la surface totale des images est supérieure à ξ.

Les mesures de similarité des approches présentées dans cette section dépendent des caractéristiques visuelles utilisées pour représenter les images. La section qui suit présente une similarité visuelle d'images organisées en Arbre Quaternaire Générique dans laquelle nous pensons que les similarités proposées dans les autres travaux de recherche peuvent s'inclure.

6.2 Arbre Quaternaire Générique et similarité d'images

La notion de Q-similarité, définie en 5.1.2, permet d'évaluer l'économie de stockage des images. Il s'agit de la distance entre les arbres quaternaires qui représentent les images. Une autre mesure de similarité, appelée *S-similarité*, peut être définie sur les images stockées dans l'Arbre Quaternaire Générique. La S-similarité correspond à un coefficient de similarité de surface des images, c'est-à-dire à une similarité visuelle. Elle est définie en 6.2.1. La comparaison entre l'Arbre Quaternaire Générique et les approches de similarité d'images, présentées ci-avant, est faite en 6.2.2.

6.2.1 Similarité de surface des images d'un Arbre Quaternaire Générique

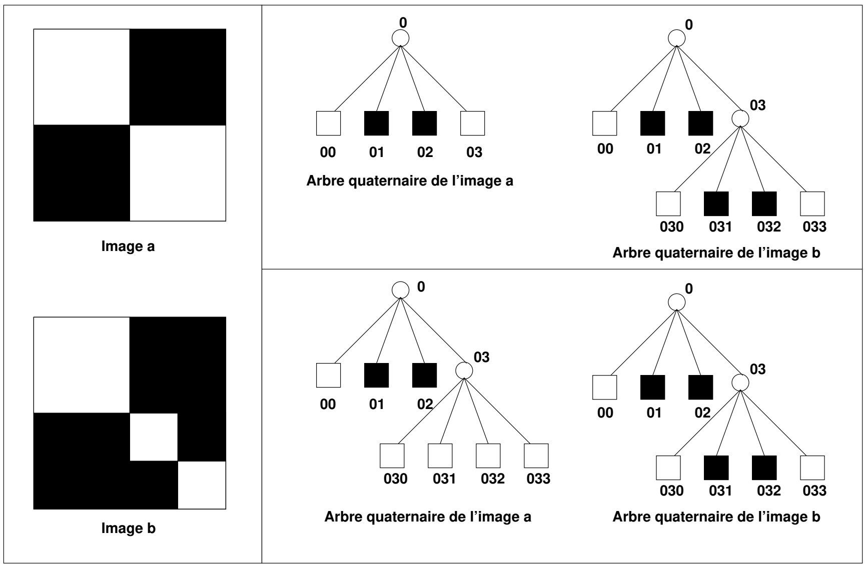


FIG. 6.3 – **L'uniformisation des arbres quaternaires de deux images dont on doit calculer le coefficient de S-similarité.**

Pour calculer le coefficient de S-similarité de deux images, les arbres quaternaires des images sont complétés ; certains nœuds peuvent être décomposés en sous-nœuds, de sorte que les arbres quaternaires aient le même nombre d'identificateurs de nœud (voir figure 6.3).

Puis, à chaque niveau p de l'arbre est associé un *coefficient de surface* de valeur $4^{-p} = 1/4^p$; plus la région est située profondément dans l'arbre quaternaire, plus le coefficient est faible. Ainsi, les quatre premiers fils de la racine de l'arbre quaternaire ont un coefficient de 4^{-1}, soit 1/4. En effet, les régions correspondantes dans l'image occupent 1/4 de la surface de l'image.

La S-similarité reprend le même principe que la Q-similarité ; il s'agit du rapport du nombre de nœuds communs (multiplié par les coefficients de surface) sur le nombre d'identificateurs de nœuds au total (sans doublon et multiplié par les coefficients de surface). A part les coefficients de surface, la différence entre la Q-similarité et la S-similarité est que seuls les nœuds feuilles sont comptabilisés pour le calcul du coefficient de S-similarité.

De manière plus formelle, on note $S_k(i,i')$, l'ensemble des nœuds feuilles, partagés par les arbres quaternaires des images i et i', ces nœuds étant situés à une profondeur k de chaque arbre quaternaire, avec $k \in [1,p]$, p étant la profondeur des arbres quaternaires des images i et i'. On note $U_k(i,i')$, l'ensemble (sans doublon) des identificateurs de nœuds feuilles apparaissant les arbres quaternaires des images i et i'. On note $\mid S_k(i,i') \mid$ (resp. $\mid U_k(i,i') \mid$) le nombre d'éléments de $S_k(i,i')$ (resp. de $U_k(i,i')$).

Le coefficient de S-similarité des images i et i', notée $s(i,i')$ est calculé par l'équation suivante :

$$s(i,i') = \frac{\sum_{k=0}^{p} \mid S_k(i,i') \mid *4^{-k}}{\sum_{k=0}^{p} \mid U_k(i,i') \mid *4^{-k}}$$

On dit que deux images i et i' sont ε S-similaires si $s(i,i') = \varepsilon$.

Exemple 6.1 Le coefficient de S-similarité des images de la figure 6.3 vaut :

$s(a,b) = \frac{3*4^{-1}+2*4^{-2}}{3*4^{-1}+4*4^{-2}} = \frac{7}{8}$

- $\mid S_1(a,b) \mid = 3$, car les nœuds feuilles 00, 01 et 02 ont même valeur dans les deux arbres quaternaires. $\mid S_2(a,b) \mid = 2$ car les nœuds feuilles 030 et 033 ont la même valeur dans les arbres quaternaires de a et b, après uniformisation des arbres quaternaires (voir le bas de la figure 6.3).
- $\mid U_1(a,b) \mid = 3$, car les nœuds 00, 01 et 02 sont les seuls nœuds feuilles situés au niveau 1 des arbres quaternaires. En revanche, $\mid U_2(a,b) \mid = 4$, car les nœuds $03x$, $x \in [0,3]$ sont tous feuilles au niveau 2 des arbres quaternaires uniformisés des images a et b.

  Etant donné le découpage des images, on déduit que les images a et b sont 7/8 S-similaires, c'est-à-dire que 7/8 de leur surface est commune.

La S-similarité correspond à une similarité visuelle des images, qui est notamment très pratique dans le cas d'images en niveau de gris. Cette similarité se calcule à partir des images organisées en Arbre Quaternaire Générique. Il est même envisageable de calculer la similarité de surface de régions d'images.

Le coefficient de S-similarité d'une région r dans deux images i et i', telle que r est située à une profondeur $p(r)$ dans les arbres quaternaires des images, a pour valeur :

$$s(i,i',r) = \frac{\sum_{k=p(r)}^{p} \mid S_k(i,i') \mid *4^{-[k-p(r)]}}{\sum_{k=p(r)}^{p} \mid U_k(i,i') \mid *4^{-[k-p(r)]}}$$

Exemple 6.2 Le calcul du coefficient de S-similarité de la région 03 dans les images a et b, stockées dans l'Arbre Quaternaire Générique de la figure 5.3 (voir page 53), se fait en plusieurs étapes :

1. La région 03 est recherchée dans l'Arbre Quaternaire Générique, et sa valeur dans les images a et b est recherchée.

2. Comme le nœud 03 est interne dans l'arbre quaternaire de b (la valeur associée à b est I) et qu'il est blanc dans l'arbre quaternaire de a, il faut comparer la valeur associée à l'image a dans le nœud générique 03 avec les valeurs associées à l'image b dans les nœuds génériques $03x$, $x \in [0,3]$. Cette comparaison peut se faire parce que seule la couleur des nœuds d'arbre quaternaire est stockée dans les nœuds générique. Si les images avaient été en niveaux de gris, il aurait fallu découper la région 03 en quatre dans l'image a et utiliser une similarité (sur un ensemble de caractéristiques donné) pour comparer les sous-régions $03x$, $x \in [0,3]$, dans les deux images.

3. Les nœuds $03x$, $x \in [0,3]$, sont tous feuilles dans l'arbre quaternaire de b, la profondeur p pour le calcul du coefficient de S-similarité de la région 03 dans les images a et b est donc de 2.
   De l'étape précédente, on déduit :

   $s(a,b,03) = \frac{\sum_{k=2}^{2} |S_k(a,b)|*4^{-[k-p(r)]}}{\sum_{k=2}^{2} |U_k(a,b)|*4^{-[k-p(r)]}} = \frac{2*4^{-(2-2)}}{4*4^{-(2-2)}} = \frac{1}{2}$

   Etant donné le découpage des images en arbre quaternaire, les images a et b sont 0.5 S-similaires pour la région 03.

Pour calculer le coefficient de S-similarité d'images, il est également possible de demander à l'utilisateur de choisir la profondeur maximale jusqu'où il souhaite que la comparaison ait lieu. Soit p' la profondeur maximale choisie par l'utilisateur pour comparer deux images i et i'. Les arbres quaternaires des images i et i' ne sont uniformisés que jusqu'à la profondeur p'. Le coefficient de S-similarité des images est dans ce cas défini par :

$$s_{p'}(i,i',r) = \frac{\sum_{k=0}^{p'} \mid S_k(i,i') \mid *4^{-k}}{\sum_{k=0}^{p'} \mid U_k(i,i') \mid *4^{-k}}$$

Exemple 6.3 Le coefficient de S-similarité des images a et b de la figure 6.3, lorsque la profondeur p' est fixée à 1, est égal à :

$s_1(a,b) = \frac{\sum_{k=0}^{1} |S_k(a,b)| * 4^{-k}}{\sum_{k=0}^{1} |U_k(a,b)| * 4^{-k}} = \frac{3*4^{-1}}{4*4^{-1}} = \frac{3}{4}$

Lorsque la profondeur des arbres quaternaires des images a et b est de niveau 1, les images a et b sont 3/4 S-similaires.

Dans les définitions et dans les exemples précédents, la comparaison de deux régions se fait de manière exacte ; les valeurs d'un même nœud n dans deux arbres quaternaires sont égales ou non. Il est évidemment possible de moduler cette comparaison en demandant à l'utilisateur de fixer un seuil ε : deux valeurs v et v' d'un même nœud sont considérées comme égales si leur différence ne dépasse pas ε, $\mid v' - v \mid < \varepsilon$. Cette opération permet de comparer des régions sans uniformiser les arbres quaternaires, en particulier lorsque les profondeurs des arbres quaternaires sont très différentes. Si, par exemple, les nœuds internes des arbres quaternaires contiennent de l'information sur les nœuds fils (un motif ou la moyenne de la valeur des nœuds fils), la S-similarité peut être calculée en comparant la valeur d'un nœud interne avec celle d'un nœud feuille.

Exemple 6.4 On suppose que le nœud interne 03 dans l'arbre quaternaire de l'image b (voir figure 6.3) contient de l'informations sur ses nœuds fils : les nœuds $03x$, $x \in [0,3]$, forment un damier noir et blanc. La similarité des régions 03, calculée sur la couleur, dans les deux images a et b indique que ces images sont S-similaires pour la moitié de la région 03 (la région occupant un quart de la surface des images). Par conséquent :

$s(a,b) = s_1(a,b) + \left(\frac{1}{2}\right) * 4^{-1} = \frac{7}{8}$

On déduit bien que les images a et b sont 7/8 S-similaires.

La S-similarité est une mesure de similarité visuelle complémentaire de la Q-similarité présentée dans le chapitre 5. Plus précisément la similarité calculée à partir de l'égalité des valeurs de nœud d'arbre quaternaire, ou Q-similarité, entraîne l'égalité des valeurs des quadrants correspondants, et donc la S-similarité. La réciproque n'est pas vraie comme le montre l'exemple suivant : soit une image i toute blanche et une image i' blanche composée d'un pixel noir. Le coefficient de Q-similarité des images i et i' est voisin de zéro. En effet, l'arbre quaternaire de i est composé d'une feuille blanche et l'arbre quaternaire de i', si l'utilisateur choisit un découpage par la couleur, peut aller jusqu'au pixel de l'image. En revanche, le coefficient de S-similarité de ces images est voisin de 1.

La sous-section qui suit compare l'Arbre Quaternaire Générique avec les approches proposant des mesures de similarités entre images.

6.2.2 Comparaison avec les approches de similarité d'images

Des mesures de similarité, très différentes de nos propositions, ont été proposées dans d'autres travaux de la recherche. Le tableau 6.1 rappelle, pour chaque approche citée dans ce mémoire, les caractéristiques à partir desquelles la similarité entre les images est calculée.

Ces mesures de similarité peuvent être intégrées à l'Arbre Quaternaire Générique pour comparer

Tab. 6.1 – **Recherche par similarité des caractéristiques d'image**

Approches	**Caractéristiques sur laquelle la similarité est calculée**
[LOT94], [FSN+95], [SC96], [Smi97], [OTCH98], [Fal99]	Couleur
[PF95], [FSN+95], [DSBM97], [MAK97], [PF97]	Forme
[NBMM98], [MBNW99], [NRS99]	Combinaison de plusieurs caractéristiques
[FSN+95], [SC96], [OTCH98], [MG98]	Localisation spatiale d'une ou plusieurs caractéristiques
[NTOL97], [OTOL+97], [NTOL99]	Caractéristiques des objets de l'image et relations spatiales entre objets
[SC94], [FSN+95], [Smi97], [SZB99]	Texture
[Smi97], [MBNW99], [NRS99]	Caractéristiques des régions d'images
[Chi94], [RKV95], [BKSS90], [WJ96], [KSF+96] [BKK96], [BO97], [CPZ97], [CF98], [MBNW99]	Représentation des caractéristiques par un point multidimensionnel

TAB. 6.2 – **Recherche d'images par similarité de régions**

Approches	**Découpage de l'image**	**Caractéristiques**	**Spécificité**
[MBNW99]	Découpage de l'image en arbre quaternaire en laissant l'utilisateur choisir la profondeur de l'arbre	Combinaison de diverses signatures	Comparaison des régions des images de la base avec des régions de l'image requête ayant subies des translations
[NRS99]	Découpage de l'image par une fenêtre de taille fixe choisie par l'utilisateur	Combinaison de caractéristiques à l'aide d'ondelettes	Utilisation de seuils pour calculer la similarité des régions et la similarité des images entières
[Smi97]	Découpage de l'image à l'aide d'une librairie de motifs	Texture et couleur	

les morceaux d'images stockées dans la base. En effet, dans les exemples présentés dans les sections 5.2.3, 5.2.6, 5.3 et 6.2, les images comparées sont en noir et blanc, afin de faciliter la compréhension du lecteur. Dès que les images sont en niveaux de gris ou que des histogrammes, par exemple, sont stockés dans les nœuds des arbres quaternaires, des mesures de similarité doivent être utilisées pour comparer les différentes valeurs de nœud. Le calcul du numérateur de la S-similarité doit se faire, lorsque les images ne sont pas en noir et blanc, à l'aide d'une mesure de similarité.

Il est également possible de représenter les caractéristiques des morceaux d'images, stockés dans l'Arbre Quaternaire Générique, par des vecteurs multidimensionnels et d'appliquer les mécanismes proposés dans les approches de requêtes de voisinage (voir section 6.1.3 et dernières ligne du tableau 6.1) pour les comparaisons.

Parmi les approches de recherche d'images par similarité de régions (voir tableau 6.2), celle de [MBNW99] est la plus proche de notre proposition. En effet, un arbre quaternaire est utilisé pour faciliter les recherches. Cependant, le découpage de l'image en arbre quaternaire se fait non pas selon un critère précis, mais selon la profondeur maximale choisie par l'utilisateur. Dans ce cas, les arbres quaternaires sont équilibrés (toutes les feuilles sont au même niveau). Dans l'Arbre Quaternaire Générique, les images sont découpées selon un critère donné (voir section 4.2.2), l'utilisateur pouvant choisir de découper uniquement certaines régions. Les arbres quaternaires des images, stockées dans un Arbre Quaternaire Générique, ne sont donc pas équilibrés. Ils sont éventuellement équilibrés en mémoire lors du calcul de la S-similarité (voir section précédente). L'Arbre Quaternaire Générique offre donc plus de possibilités pour comparer les images.

Comme dans les approches de [MBNW99, NRS99], l'utilisateur peut choisir la profondeur des arbres quaternaires des images pour calculer leur S-similarité, ainsi que le seuil de similarité des régions (ε). En outre, dans l'Arbre Quaternaire Générique, il est possible de naviguer dans une même région de différentes images, puisqu'un nœud générique contient toute l'information de la région correspondante dans toutes les images de la base.

Comme l'arbre quaternaire multi-niveaux de [LOT94] (voir section a)), l'Arbre Quaternaire peut être utilisé pour faire des comparaisons d'images par niveaux. Il est même possible de faire des comparaisons à différents niveaux par région (comparer la région 01 jusqu'au niveau 3 mais la région 02 jusqu'au niveau 1, par exemple). L'Arbre Quaternaire offre donc plus de modularité que les autres approches. Il peut par conséquent tout à fait s'intégrer dans un système de recherche d'images par le contenu, en utilisant les mesures de similarité proposées dans les autres travaux de recherche.

Les approches de recherche d'image par le contenu ne s'intéressent pas au stockage des images. Les images sont stockées indépendamment les unes des autres, des méta-informations sur ces images pouvant être contenues dans une base de données. L'optimisation du stockage n'est pas un objectif pour ces approches. Dans la section suivante, l'Arbre Quaternaire Générique est comparé avec des approches dont l'objectif majeur est l'optimisation du stockage de séquences d'images.

6.3 Comparaison avec le chevauchement d'arbres quaternaires

Les approches de chevauchement, proposées dans [VME93, VMK95], sont orientées vers la gestion de séquences d'images similaires (ex. séquences d'images vidéo), alors que l'Arbre Quaternaire Générique est plus orienté par le traitement ou la conception d'images. Ces propositions sont néanmoins les plus proches de nos objectifs : elles optimisent le stockage, mais restreignent, voire n'autorisent pas, certaines opérations. La comparaison de l'Arbre Quaternaire Générique avec ces approches est réalisée en quatre points : [1] l'organisation des images et partage d'information (en 6.3.1), [2] l'espace occupé par les images (en 6.3.2), [3] l'accès aux donnés (en 6.3.3) et [4] les opérations (en 6.3.4).

La tableau 6.3 rappelle les caractéristiques de différentes approches de chevauchement, décrites dans le chapitre 4. Les approches [VME93, VMK95] permettent la gestion de tout type d'images (binaire, en niveaux de gris, etc.). En revanche, les approches utilisant des arbres quaternaires linéaires [Lin97, TVM98] sont proposées pour gérer uniquement des images en noir et blanc. L'arbre Quaternaire n'est donc pas comparé à ces approches. Le tableau 6.4 compare l'Arbre Quaternaire Générique avec l'approche de [VME93].

6.3.1 Organisation des images et partage d'information

Dans les approches de chevauchement, l'ensemble des images est organisé en séquence. Le partage de sous-arbres quaternaires est toujours implicite et n'est possible qu'avec l'arbre quaternaire

TAB. 6.3 – **Récapitulatif des approches de chevauchement d'arbres quaternaires**

Approches	Description
[VME93]	Chevauchement d'arbres quaternaires implantés de manière hiérarchique. **Objectif :** Optimiser le stockage de séquences d'images similaires (ex. séquences d'images vidéo). Application des mécanismes de stockage différentiel aux arbres quaternaires : copie de tout un chemin de la racine à un nœud feuille et partage de nœuds avec l'arbre quaternaire de l'image précédente dans la séquence.
[TVM98]	Reprise de [VME93] en appliquant les mécanismes de chevauchement à des arbres B+. **Objectif** : Transposer l'approche de [VME93] au stockage linéaire d'arbre quaternaire. Les identificateurs de nœuds d'arbre quaternaire de couleur noire sont insérés dans un arbre B+. L'approche est donc limitée à la gestion des images en noir et blanc. A chaque arbre quaternaire correspond un arbre B+. Le chevauchement est appliqué aux arbres B+.
[Lin97]	Chevauchement d'arbres quaternaires implantés linéairement. **Objectif :** Optimiser le stockage de séquences d'images en noir et blanc similaires (ex. séquence d'images vidéo). Codage des images par un code linéaire composé de chiffres (0,1,2 et 3). Codage des différences deux à deux entre images de la séquence par un code composé de lettres (A, B, C et D). La lecture d'une image impose un décodage de la séquence depuis la première ou la dernière image.

TAB. 6.4 – **Comparaison de l'Arbre Quaternaire Générique avec le chevauchement**

Chevauchement [VME93]	**Arbre Quaternaire Générique**
Création d'une image supplémentaire à chaque groupe de mises à jour	Mise à jour directe de n'importe quelle image
Copie de tout un chemin de l'arbre quaternaire à chaque ajout d'image	Pas de redondance
Pas de suppression d'images	Suppression logique / physique
Lecture de plusieurs chemins dans les arbres quaternaires pour accéder à une même région dans différentes images	Accès direct à une même région dans différentes images
Approche orientée vers le stockage de séquences d'images non modifiables (séquence d'images vidéo)	Approche orientée vers le stockage d'images dans un atelier de traitement d'images

immédiatement précédent dans la séquence. Par conséquent, il ne peut pas y avoir de partage lorsqu'une valeur de quadrant apparaît plusieurs fois de manière périodique ou non.

En revanche, l'Arbre Quaternaire Générique maximise le partage de quadrants entre arbres quaternaires :

- Quand une même valeur apparaît dans un nœud n de deux arbres quaternaires différents, elle est automatiquement partagée, implicitement ou explicitement.
- L'organisation des images dans l'Arbre d'Images permet de maximiser le partage implicite, puisqu'une image insérée dans l'Arbre d'Images comme fille de l'image dont elle est le plus Q-similaire (c'est-à-dire dont l'arbre quaternaire est le plus proche).

Plus précisément, comme il est expliqué à la fin du chapitre 5, le nombre maximum de lignes dans un nœud générique n correspond au nombre de valeurs différentes prises par le nœud n dans les arbres quaternaires des images de la base, auxquelles s'ajoutent les valeur Int (nœud interne) et $\perp$ (*n'existe pas*). Dans le cas d'images en noir en blanc, le nombre de lignes par nœud générique est au maximum de quatre. L'organisation des images en Arbre d'Images permet d'augmenter le partage implicite et donc de diminuer la taille des nœuds génériques.

Par conséquent, l'Arbre Quaternaire a tendance à moins grossir qu'une structure d'arbres quaternaires chevauchés, notamment lorsque le nombre d'images stockées est important. Le partage des valeurs de nœud est donc équivalent, dans le pire des cas, entre le chevauchement d'arbres quaternaires et l'Arbre Quaternaire Générique. En général, le partage est meilleur dans le cas de l'Arbre Quaternaire Générique.

6.3.2 Optimisation de l'espace occupé par les images

Dans les approches de chevauchement, la suppression est impossible et seule la dernière image de la séquence peut être modifiée. L'insertion d'une image à l'intérieur de la séquence est interdite. Une nouvelle image est toujours insérée à la fin de la séquence. Par conséquent, la mise à jour de l'espace occupé par une structure d'arbres quaternaires chevauchés ne peut se faire que par l'insertion d'une nouvelle image à la fin de la séquence. Dans le cas où une partie de la nouvelle image est non partageable avec la dernière image de la séquence, un nouveau sous-arbre est construit à partir de la racine (voir chapitre 4). Plus précisément, lorsque deux arbres quaternaires se chevauchent, il y a au minimum l'insertion d'un nouveau chemin de la racine à un nœud feuille. La profondeur minimale d'un arbre quaternaire est de l'ordre de $log_4(N)$ où N correspond au nombre de nœuds dans l'arbre. Par conséquent, l'espace supplémentaire occupé par une structure de k arbres quaternaires chevauchés est au minimum de $k * log_4(N)$ nœuds.

Dans l'Arbre Quaternaire Générique, l'espace occupé par les images peut être modifié par l'insertion, la modification ou la suppression d'images. La création ou la suppression d'une valeur de nœud peut être assimilée à une modification. L'attention est donc focalisée sur la modification (voir chapitre 5, section 5.2.2).

Lorsque la valeur v de nœud n est modifiée (ou créée) dans l'arbre quaternaire d'une image i, trois cas apparaissent :

1. Le nœud n n'existe pas dans l'Arbre Quaternaire Générique, parce que n n'apparaît dans aucun arbre quaternaire d'image de la base avant la mise à jour de i.

   $\Longrightarrow$ Le nœud générique n est créé. Il contient deux lignes. Une ligne associe la valeur v à l'image i. Comme le nœud n n'existe que dans l'arbre quaternaire de l'image i, l'autre ligne associe la valeur $\perp$ (*n'existe pas*) à l'image racine de l'Arbre d'Images et aux éventuelles images filles de i lorsque le partage implicite entre i et ses filles a du être supprimé.

2. Le nœud n existe dans l'Arbre Quaternaire Générique mais la valeur v de n dans l'arbre quaternaire de i n'existe pas dans le nœud générique n.

   $\Longrightarrow$ Une nouvelle ligne est par conséquent ajoutée au nœud générique n. Cette ligne associe la valeur v à l'image i. Le partage implicite entre i et ses descendants est supprimé.

3. Le nœud n existe dans l'Arbre Quaternaire Générique et la valeur existe dans les arbres quaternaires d'autres images de la base :

   (a) L'identificateur de l'image i est associé à la valeur v dans la ligne correspondante du nœud générique n, si le partage est implicite. Le partage implicite entre l'image i et ses descendants est supprimé.

   (b) Rien n'est fait si l'image i peut être associée implicitement à la valeur v du nœud générique n.

Par conséquent, l'Arbre Quaternaire Générique est toujours meilleur du point de vue de l'espace occupé par les images que les approches de chevauchement. Plus précisément, l'Arbre Quaternaire Générique est plus performant, sur l'espace occupé, pour les raisons suivantes :

- Soit k le nombre d'éléments de l'ensemble initial d'images stockées dans la structure d'arbres quaternaires chevauchés et dans l'Arbre Quaternaire Générique. Après un nombre m de groupes de mises à jour réalisées sur les images, le nombre d'images stockées dans l'Arbre Quaternaire Générique est toujours de k. En revanche, comme chaque groupe de mises à jour sur la structure d'arbres quaternaires chevauchés crée un nouvel arbre quaternaire et donc une nouvelle image, le nombre d'images après les m mises à jour est de $m + k$ dans la structure chevauchée.

- Dans les approches de chevauchement, un arbre quaternaire partage des nœuds avec l'arbre quaternaire immédiatement précédent dans la séquence. En revanche, dans l'Arbre Quaternaire Générique, tous les arbres quaternaires des images stockées peuvent partager explicitement ou implicitement des valeurs de nœud.

6.3.3 Accès aux images

Trois cas d'accès aux images sont distingués : [1] l'accès à une image entière, [2] l'accès à une région d'images et [3] l'accès à un ensemble d'images.

a) Accès à une image entière

Il existe deux types de valeurs de nœuds feuilles dans les nœuds génériques. Lorsque la valeur est petite, par exemple lorsqu'il s'agit de la couleur des quadrants et que le découpage des images s'est fait par homogénéité, la valeur est stockée dans les nœuds génériques. Lorsque la valeur est trop grande pour être stockée dans les nœuds génériques, dans le cas où la valeur est un quadrant de l'image, comme par exemple dans l'application médicale décrite dans le chapitre 3 (voir figure 5.17 p. 76), le nœud contient un pointeur vers l'espace mémoire ou le fichier contenant la valeur du nœud. Par conséquent, l'Arbre Quaternaire Générique, comme la structure d'arbres quaternaires chevauchés peut tenir en mémoire. La différence en terme de performances d'accès aux images concerne seulement les opérations en mémoires principale.

La lecture d'une image i dans une séquence d'arbres quaternaires chevauchés est réalisée en deux étapes :

1. La racine de l'arbre quaternaire de l'image i est recherchée. La complexité de cette opération est en $\mathcal{O}(k')$, avec k' le nombre d'images stockées.

2. Les nœuds de l'arbre quaternaire de l'image i sont lus, depuis le nœud racine.

Si l'image i est stockée dans l'Arbre Quaternaire Générique, sa lecture est réalisée en lisant le même nombre de nœuds que la structure d'arbres quaternaires chevauchés. Cependant, chaque nœud est un nœud générique. La complexité de lecture d'un tel nœud (voir section 5.2.1) est $\mathcal{O}(k)$, avec k le nombre d'images stockées dans l'Arbre Quaternaire Générique. Comme il a été expliqué précédemment en 6.3.2, $k' = k + m$ avec $m \geq 0$, m pouvant être très grand par rapport à k.

b) Accès à une région d'image

Parmi les représentations d'images similaires par des arbres quaternaires, l'Arbre Quaternaire Générique est spécifique dans le sens où les nœuds génériques peuvent être utilisés selon deux dimensions. La première dimension permet de naviguer à travers l'arbre quaternaire d'une image : il s'agit d'une structure hiérarchique. L'autre dimension consiste à passer d'une image à l'autre à travers un même nœud générique. Il est par conséquent facile, avec l'Arbre Quaternaire Générique, d'accéder simultanément à une même région dans plusieurs images. Le mécanisme de partage intégré dans l'Arbre Quaternaire Générique facilite donc la comparaison des valeurs d'un nœud d'arbre quaternaire.

De plus, l'accès à une partie d'image ou à une partie dans plusieurs images est réalisable directement en accédant au nœud générique correspondant à la région. En revanche, dans la structure d'arbres quaternaires chevauchés, l'accès à un nœud se fait toujours depuis la racine de chaque arbre quaternaire. La comparaison d'un nœud n dans différents arbres quaternaires impose la lecture d'un chemin dans chaque arbre quaternaire.

c) Accès à un ensemble d'images

L'implantation linéaire [Lin97] impose un ordre linéaire de lecture et a pour conséquence, à la lecture d'une image située à l'intérieur de la séquence, une reconstruction complexe de l'image (voir chapitre 4, section 4.3.2). La modification de l'ordre des images impose la reconstruction de tout le codage linéaire de la séquence. Les utilisateurs sont donc obligés de lire les images dans l'ordre dans lequel elles ont été stockées. Cette contrainte peut être levée dans le cas de l'implantation hiérarchique des arbres quaternaires, comme dans [VME93] et dans l'Arbre Quaternaire Générique.

6.3.4 Opérations

Les seules opérations possibles sur une structure d'arbres quaternaires chevauchés sont la lecture d'une image et l'insertion d'une image à la fin de la séquence. Il est également possible de modifier, voire supprimer la dernière image de la séquence.

En ce qui concerne les images stockées dans un Arbre Quaternaire Générique, il n'existe aucune limitation sur les opérations applicables aux images : une image peut être insérée comme fille de n'importe quelle image de la base, il est possible de supprimer ou de mettre à jour n'importe quelle image de la base.

Dans l'Arbre Quaternaire Générique, les images sont organisées en Arbre d'Images afin d'augmenter le partage et donc d'optimiser l'espace occupé par les images. Mais les utilisateurs peuvent à tout moment créer leur propre séquences d'images ou ordonner les images selon leurs besoins, indépendamment de l'Arbre d'Images (voir section 5.1.5).

La différence la plus importante entre l'Arbre Quaternaire Générique et les approches de chevauchement est que l'Arbre Quaternaire Générique rend faisable et/ou facilite plusieurs opérations n'existant pas dans les structures d'arbres quaternaires chevauchés : la mise à jour d'une image quelconque, la comparaison d'images et le suivi d'une région dans plusieurs images. Cette structure est donc très utile dans des applications où il est nécessaire de travailler sur une image ou sur plusieurs images simultanément.

Conclusion

L'Arbre Quaternaire Générique a été proposé, dans un premier temps, pour optimiser le stockage d'images dans un atelier de traitements d'images. Les images obtenues après traitement peuvent être considérées comme des versions d'image. A notre connaissance, peu d'approches [KTKM96, ATF+97] proposent des mécanismes de gestion de versions d'image. Une version d'image correspond, dans [KTKM96, ATF+97], à une image résultant de l'application d'un algorithme de traitement sur une image initiale. Chaque version d'une image i est associée à un objet spécifique qui regroupe toutes les versions de l'image i. Les images et leurs versions sont stockées dans une base de données de sorte que les utilisateurs puissent rechercher des algorithmes de traitement particulier, extraire des versions d'image pour appliquer de nouveaux algorithmes, etc. Cependant, les auteurs de [KTKM96, ATF+97] gèrent des versions d'image comme un tout, c'est-à-dire comme des images entières. Ils ne tiennent par conséquent pas compte de l'espace occupé pour stocker les images.

L'objectif de l'Arbre Quaternaire Générique est double : cette structure doit non seulement optimiser le stockage des images mais également permettre aux utilisateurs d'appliquer des traitements sur les images ou de les comparer.

Il existe deux grandes catégories d'approches, celles concernées par le stockage des images, dont font partie les approches de chevauchement, et celles axées vers la recherche d'images par le contenu. L'Arbre Quaternaire Générique est une structure plus complète que les structures d'arbres quaternaires chevauchés : l'optimisation du stockage est équivalente ou meilleure et l'Arbre Quaternaire Générique ne limite pas les opérations réalisables sur les images. Par rapport aux approches de similarité d'images, l'Arbre Quaternaire Générique est complémentaire. En effet, avec la notion de S-similarité, il permet de comparer les images d'un point de vue visuel, en utilisant les mesures de similarités proposés dans les autres travaux de recherche. De plus, ils permet de moduler les requêtes de similarités et facilite en particulier la recherche d'images par similarité de régions. Le domaine de l'Arbre Quaternaire Générique ne se limite pas par conséquent au traitement d'images, mais peut s'appliquer à divers domaines comme les applications géographiques ou la réalité virtuelle, c'est-à-dire des applications où il peut être nécessaire d'opérer sur les images mais où les recherches d'images occupent également une part importante.

L'arbre quaternaire est une structure non équilibrée qui croît par les feuilles, la racine de l'arbre demeurant toujours la même. Il est utilisé principalement dans le domaine des images et dans certains cas dans le domaine géographique lorsque les requêtes portent sur des régions fixes [PRS95, SVP+96]. D'autres structures, comme les arbres B+ ou les arbres R, sont très utilisées dans le domaine des bases de données et permettent également de gérer des objets de grande taille (voir chapitre 1). La partie C, qui suit, présente une généralisation des mécanismes de l'Arbre Quaternaire Générique aux autres structures arborescentes, notamment lorsque ces structures sont équilibrées et qu'elles croissent par le haut. Les mécanismes de gestion des images sont donc étendus aux objets de grande taille, en général.