UNIVERSITE des Sciences de CAEN

GREYC Equipe Image de Caen

DEA Electronique Systèmes Capteurs Images
****

Rapport de stage
2003-2004

Indexation d'images et recherche par points d'intérêt

GROUPE DE RECHERCHE EN INFORMATIQUE,
IMAGE ET INSTRUMENTATION DE CAEN
****

Présentée par :

- ✓ **Thomas Reuste**

Responsables :

- ✓ **Mr : Olivier Lézoray**
- ✓ **Mr : Régis Clouard**

Remerciements:

Je tiens d’abord à remercier M.Olivier Lézoray, mon maître de stage, pour son aide, sa disponibilité et ces conseils tout au long de mon stage.

Je tiens également à remercier M.Régis Clouard pour m’avoir permit de réaliser mon stage au sein de l’équipe image du GREYC, et pour son encadrement durant mon stage.

Enfin je remercie tous les membres de l’équipe image du GREYC ainsi que tous les thésards.

Sommaire:

Introduction

Les nouveaux supports informatiques permettent de nos jours de manipuler des quantités très importantes d'informations. Parmi ces informations, les images prennent une part de plus en plus grande. De nombreuses applications manipulent de manière quotidienne des images (multimédia, infographie, médicale, etc.). Il apparaît donc aujourd'hui essentiel de concevoir des systèmes de tri automatique (indexation), capables de mener à terme la recherche d'information au sein de gigantesques collections d'images numériques. Or, la quantité des données disponibles augmente exponentiellement et leur gestion manuelle devient impossible.

Le but des systèmes de tri automatique (indexation) est de permettre à un utilisateur de trouver, dans des bases d'images, toutes celles qui sont semblables à une image qui l'intéresse. Un programme d'indexation se conçoit comme un système qui prend en entrée une image de référence et qui retourne un critère de similarité entre l'image de référence et toutes les images de la base. Ceci permet de les trier de la plus similaire à la moins similaire.

L'intérêt de l'indexation d'image est de permettre à une personne (exemple : journaliste) qui cherche à illustrer son article (ou son reportage) par une image, d'avoir à la sortie du système d'indexation une série d'images, trier par ordre de ressemblance, en réponse à son image de référence.

Dans ce rapport, nous présentons une méthode d'indexation d'images. Ce rapport se compose de quatre parties, la première consiste en une position du problème, la deuxième portera sur la détection des points d'intérêt, la troisième sur la segmentation fine des images et la dernière sur l'indexation d'images.

CHAPITRE 1:
POSITION DU PROBLEME

1.1 L'indexation

Dans la première génération des systèmes d'indexation, les images étaient représentées par des termes sémantiques (mots-clés). Nous pouvons citer comme exemple Google.

Puis dans la deuxième génération des systèmes d'indexation, il y a eu l'intégration de différentes propriétés liées aux images :

- ► Propriétés perceptuelles : couleur, texture, forme, relations spatiales,
- ► Propriétés sémantiques : objets, scènes,
- ► Impression visuelle, signification : combinaison des deux autres.

Le but de l'indexation est de réordonner les images d'une base de données en fonction d'une distance de similarité par rapport à une image de référence. Nous pouvons réaliser la requête de similarité entre les images de différentes façons :

1) La requête peut se faire sur toute l'image ou bien sur une partie de l'image.

2) La requête peut utiliser les propriétés spatiales des objets présents dans l'image (graphe d'adjacence des régions).

Il existe différentes méthodes pour réaliser l'indexation d'images [21], soit en réalisant une approche structurelle [2], l'image est vu comme un ensemble de régions spatialement ordonnées (cf. Figure A), soit en réalisant une approche statistique, dans ce cas on utilise des statistiques globales de l'image (histogramme [22] par exemple) (cf. Figure B). De plus, il existe deux méthodes différentes pour gérer la base d'images, la première méthode est dite directe (cf. Figure C) et la deuxième est dite indirecte (cf. Figure D). Dans la méthode directe, la base d'images n'aura aucune organisation particulière. Alors que dans la méthode indirecte, la base d'images va être indexée pour avoir une première comparaison des images de la base entre elles, avant même de réaliser la comparaison avec une image de référence. Cette dernière méthode permet d'accélérer la mise en correspondance.

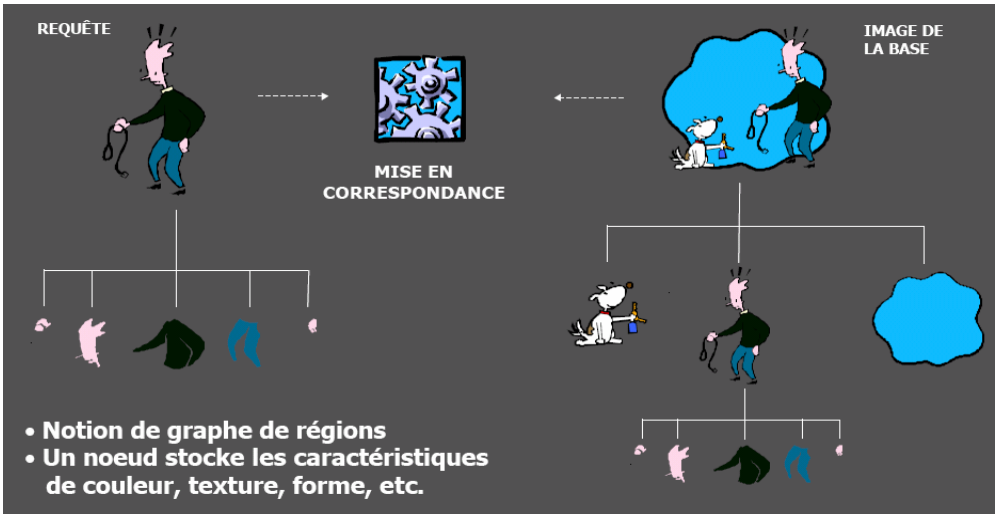


Figure A : Indexation réalisée par une approche structurelle.

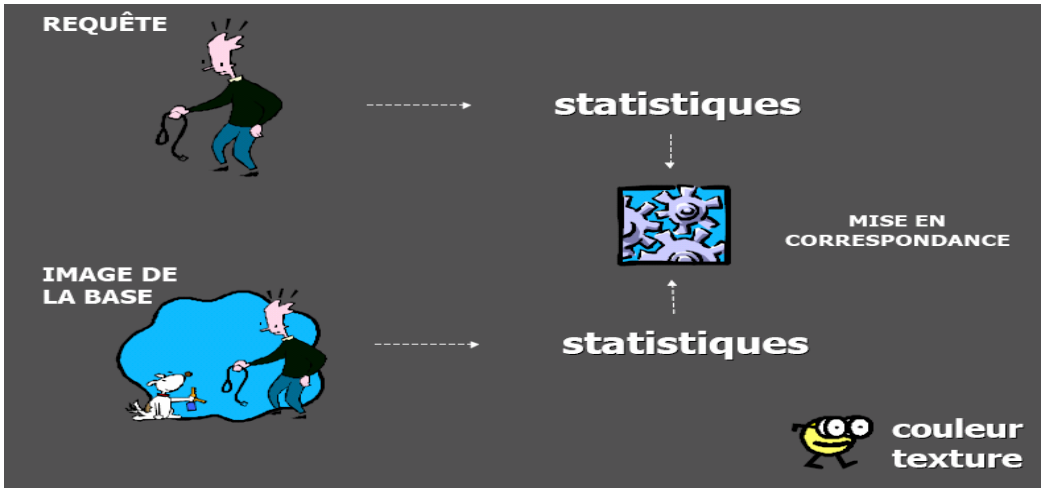


Figure B : Indexation réalisée par une approche statistique.

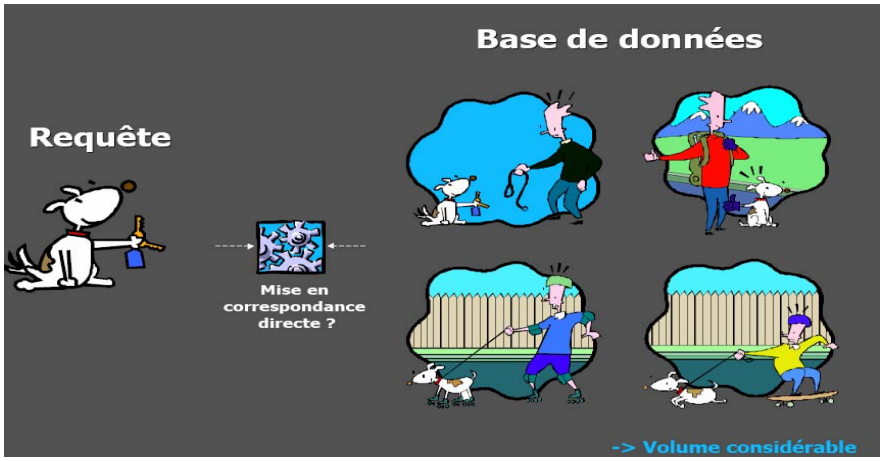


Figure C : Indexation réalisée par une mise en correspondance directe.

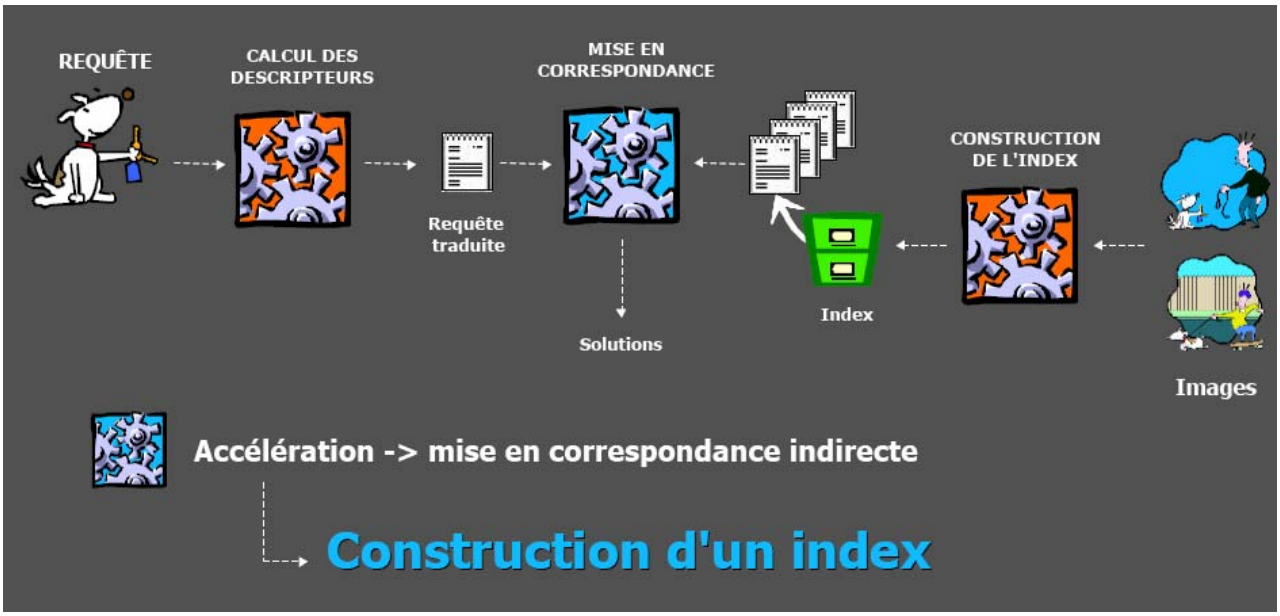


Figure D : Indexation réalisée par une mise en correspondance indirecte.

1.2 La méthode d'indexation

Dans notre projet, nous cherchons à concevoir un système d'indexation d'images (cf. Figure E). Pour réaliser l'indexation d'images, il existe différentes méthodes de mise en correspondance.

Ces méthodes peuvent être basées soit sur des descripteurs globaux (histogrammes), soit sur des descripteurs hiérarchiques (mise en correspondance de graphes), soit sur des descripteurs locaux (mise en correspondance par points d'intérêt). Pour les deux dernières méthodes, la mise en œuvre est identique, recherche puis comparaison des différentes caractéristiques des images.

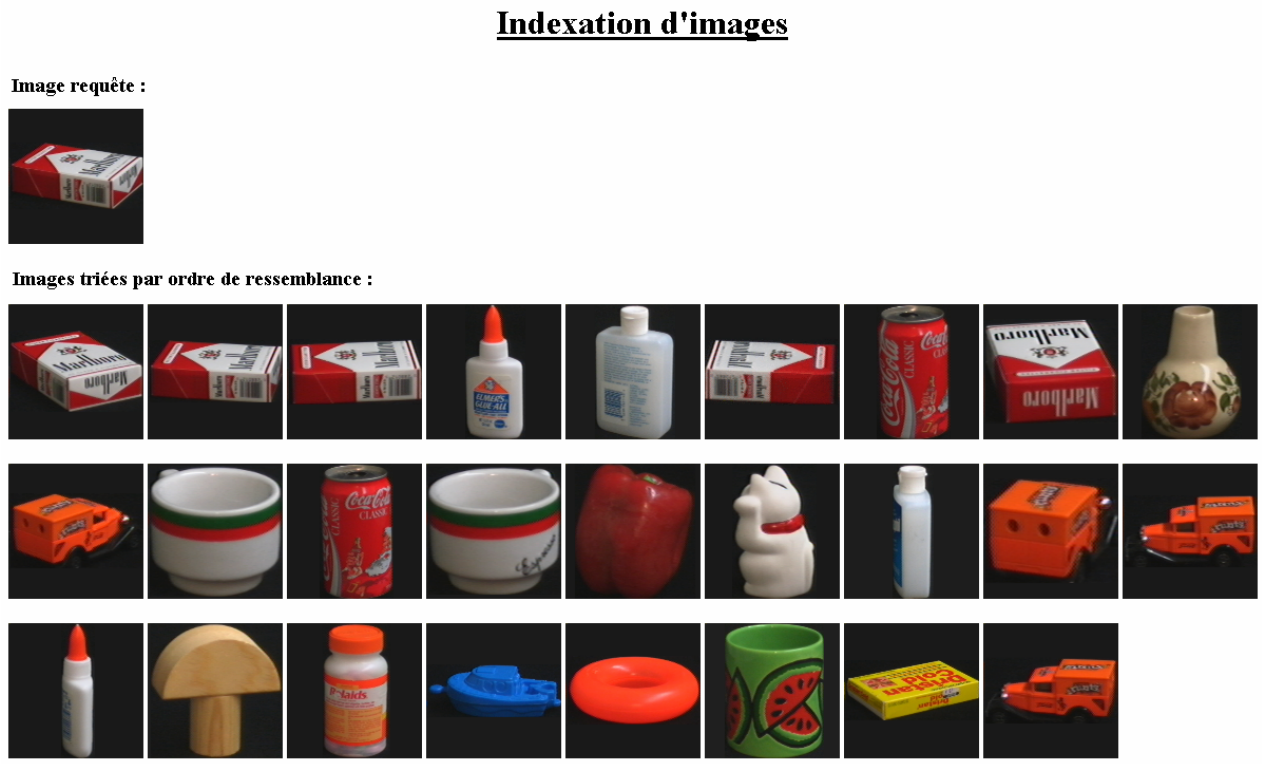


Figure E : Exemple d'indexation d'images

Le but du stage est de concevoir une méthode d'indexation d'images rapide et stable. Nous avons conçu une méthode d'indexation mixte utilisant une description locale et hiérarchique des images. Cette méthode d'indexation se base sur une mise en correspondance des points d'intérêt, ainsi que de leurs différentes caractéristiques, et sur une mise en correspondance des régions des images de la base avec celle d'une image de référence. La raison pour laquelle nous avons réalisé une méthode mixte est que la mise en correspondance par points d'intérêt peut prendre énormément de temps de calcul si les deux images à indexer comportent beaucoup de points d'intérêt. En effet il faut comparer chaque point d'une image avec tous les points de l'autre image. Alors que la mise en correspondance de graphes est plus rapide, si nous avons réalisé au préalable une segmentation fine

de l'image.

1.3 La base d'images

Dans une base de données, il existe une multitude de types d'images (IRM, microscopie médicale, dessin, etc.) Pour limiter notre étude, nous ne considérons dans ce stage que les images couleur de scènes naturelles. Ces images de scènes naturelles sont considérées comme acquises par une caméra CCD, ce qui implique un bruit gaussien de moyenne nulle et de faible écart type. De plus les images peuvent être comprimées au format jpeg, ce qui provoque des effets de pixellisation en bloc qui seront gênants lors de la détection des points d'intérêt (cf. Figure F).

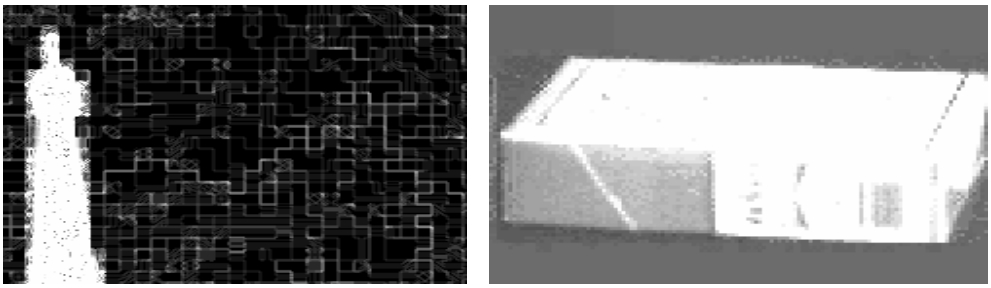


Figure F : Images illustrant les problèmes de pixellisation et de codage des images.

Pour la réalisation de ce stage, nous avons utilisé la librairie PANDORE [23,24,25] du GREYC, qui permet de programmer des traitements d'images.

CHAPITRE 2:
DETECTION DES POINTS D'INTERET

Le but du stage a consisté à implémenter une méthode permettant de détecter et d'évaluer correctement, ceci en ayant une bonne répétabilité, des points d'intérêt sur des images couleur.

Il existe déjà un opérateur dans PANDORE permettant la détection des points d'intérêt des images couleur, c'est le détecteur de Harris [4]. Les points d'intérêt détectés par le détecteur de Harris sont soit des coins en L, des jonctions en T ou en Y, des points de forte variation de texture. Ces points correspondent à des doubles discontinuités de la fonction d'intensité. Le détecteur de Harris est une référence dans l'extraction des points d'intérêt, du fait de la robustesse de son algorithme vis à vis des transformations que peuvent subir les images (rotation, translation, illumination, bruit, changement d'échelle).

Le principe de l'algorithme du détecteur de Harris est de calculer une matrice de covariance C(x,y) :

$$C(x,y) = \begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix}$$

où I_{xx}, I_{yy} et I_{xy} sont respectivement la somme des valeurs carrées de gradient en X, en Y et en X et Y dans un voisinage (2*halsize+1)* (2*halsize+1) autour du pixel (x,y).

Si la plus petite valeur propre de cette matrice au point p est positive alors ce point est considéré comme un point d'intérêt.

Pour éviter de calculer les valeurs propres, Harris propose de calculer la fonction de réponse R(x,y) pour chaque pixel par :

$$R = I_{xx} * I_{yy} - I_{xy}^{\ 2} - \text{kappa} * (I_{xx} + I_{yy})^2$$

puis de rechercher les maxima locaux de la fonction R.

Les deux paramètres d'entrée de l'algorithme de Harris sont :

- ➔ sigma qui est l'écart type de la gaussienne, et qui donne aussi la taille de la zone de recherche du maximum local (largeur = 2*halsize = 6*sigma) ;
- ➔ kappa qui est le facteur de pondération maximum pour que R soit positif.

Le détecteur de Harris [4] permet de trouver une multitude de points d'intérêt (cf. Figure 1), mais ces points n'ont pas tous la même pertinence pour la description de l'image. De plus nous avons pu remarquer du fait du codage de certaines images (en jpeg), que les régions ne sont jamais vraiment uniformes. Cela entraîne la localisation de points dans des régions qui sont théoriquement uniformes (dans la Figure 1, cela correspond au points détectés dans le fond).

Donc dans cette partie, nous avons implémenté deux opérateurs, utilisant le détecteur de Harris, qui vont permettre de résoudre ces problèmes.

Figure 1 : Image de départ et points d'intérêt trouvés avec le détecteur de Harris superposés sur l'image de départ

2.1 La pyramide Gaussienne couleur

La pyramide gaussienne couleur [4] a été décrite comme un outil puissant pour le traitement d'image. L'idée à la base de la structure pyramidale est de produire un tas d'images corrélées avec une diminution progressive de la résolution. Le but de cette diminution est d'éliminer progressivement les hautes fréquences.

La construction de chaque niveau de la pyramide gaussienne couleur [4] (cf. Figure 2) se résume en

deux étapes :

- ➔ Une convolution gaussienne de l'image, avec une valeur de sigma égale à 1 ;
- ➔ Une réduction de la résolution de l'image (dans notre stage, nous avons pris un coefficient de réduction de la résolution égale à 0.5).

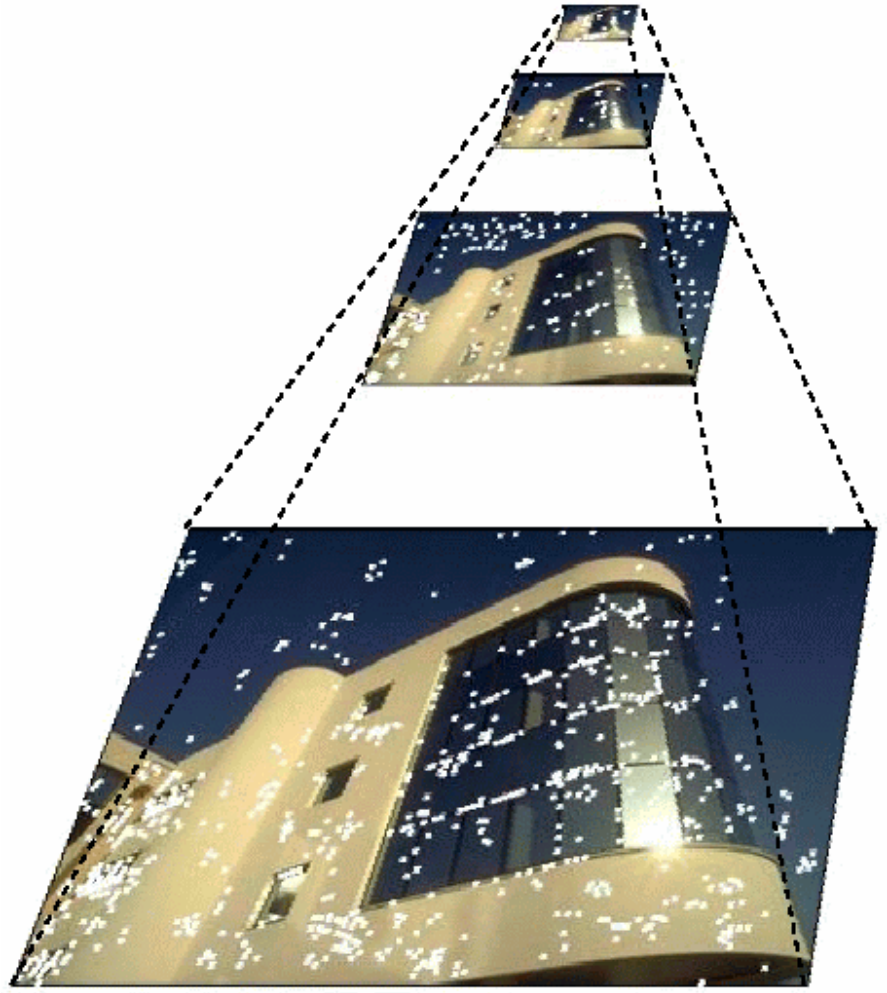


Figure 2 : Exemple de pyramide Gaussienne couleur avec les points d'intérêt superposés sur les images.

L'intérêt de cette construction est d'avoir une image couleur réelle pour chaque niveau de la pyramide, et non trois pyramides distinctes pour chaque dimension de l'espace couleur.

Une fois la pyramide [4] construite, nous passons à la détection des points d'intérêt. Pour cela

nous allons procéder en deux étapes. Dans une première étape, nous allons localiser les points d'intérêt sur toutes les images constituant la pyramide, grâce au détecteur de Harris [4].
Puis nous allons déterminer la pertinence des points trouvés par le détecteur. Pour faire ceci, nous allons nous servir de la pyramide. Si un point est pertinent pour la description de l'image, il doit se retrouver à tous les niveaux de la pyramide. Donc pour déterminer la pertinence des points d'intérêt trouvés, nous allons pour chaque point se trouvant à la base de la pyramide (image de résolution la plus élevée) regarder s'il persiste aux niveaux supérieurs de la pyramide.

Cependant, à cause de la convolution gaussienne, au fur et à mesure que nous remontons dans la pyramide, les positions des points d'intérêt varient de plus en plus, ce qui peut causer des problèmes pour retrouver un point au niveau supérieur de la pyramide. De plus, dans le cas où deux points seraient proches à la base de la pyramide, nous pourrions les confondre lors de la détection dans les niveaux supérieurs, ce qui pourrait avoir une grande influence, car un point qui aurait une faible pertinence pourrait être perçu comme ayant une bonne pertinence si nous détections le point voisin dans le niveau supérieur de la pyramide.

Du fait des problèmes de localisation des points d'intérêt dans la méthode pyramidale, nous sommes passés à une autre méthode.

2.2 Analyse Multi-Echelle

L'idée de cette méthode est la même que dans la méthode pyramidale [4]. Nous allons construire un tas d'images corrélées, sauf que la relation entre deux images se trouvant à deux niveaux différents du tas ne sera pas une diminution de la résolution, mais une augmentation du filtrage des images. Ainsi nous allons obtenir une suite d'images, de résolutions identiques, de plus en plus filtrées (schéma multi-échelle) (cf. Figure 3a). Le filtrage utilisé est la convolution gaussienne avec un sigma de 1, comme dans la méthode pyramidale.

Figure 3a : Exemple de schéma multi-échelle.

La façon de déterminer la pertinence des points d’intérêt est la même que dans la méthode précédente. Pour chaque point de l’image la moins filtrée, nous allons regarder si nous le retrouvons dans l’image suivante. Mais dans cette méthode, nous avons rajouté une mesure sur la pertinence des points d’intérêt. Plus le point se retrouve dans la suite d’images, plus sa pertinence sera grande.

Cependant nous retrouvons le même problème de localisation que dans la méthode pyramidale [4], du fait de la convolution gaussienne. Afin d’y remédier, nous avons utilisé un autre filtre qui est la diffusion couleur (filtre non linéaire) pour la construction de la suite d’images. Car ce filtre non linéaire a l’avantage de filtrer l’image, mais sans modifier les contours. Donc si nous ne modifions pas les contours, nous ne modifierons pas l’emplacement des points d’intérêt.

Avec cet opérateur, nous avons implémenté une meilleur méthode afin de localiser et de mesurer la pertinence des points d’intérêt trouvés par le détecteur de Harris [4] (cf. Figure 3b).

Cependant une bonne localisation des points d’intérêt et une mesure de pertinence ne suffissent pas pour décrire approximativement une image entière. Il faut trouver un moyen supplémentaire de décrire les points afin de pouvoir s’en servir pour l’indexation. Une méthode reconnue pour caractériser les points d’intérêt est les invariants différentiels couleur [5,7].

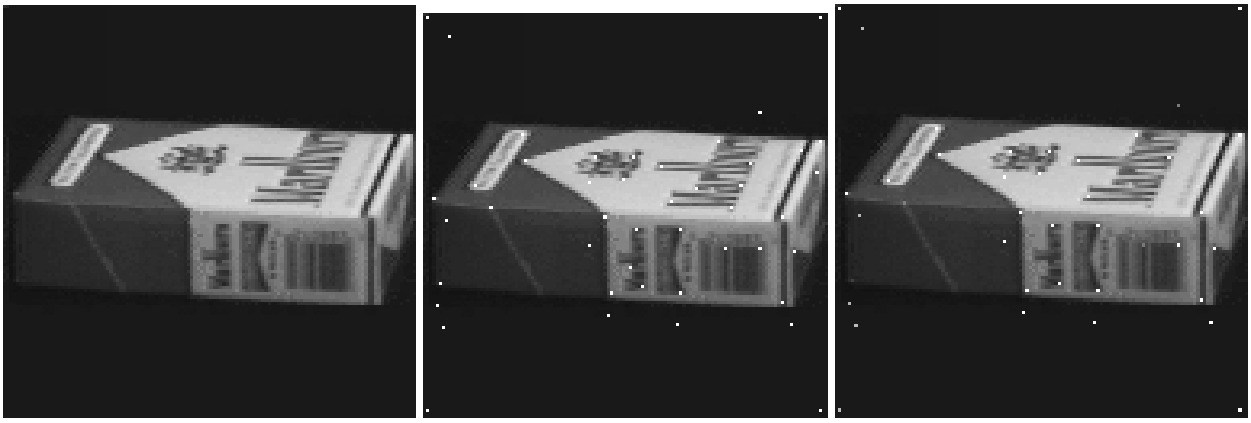


Figure 3b : Image de départ, points d'intérêt trouvés avec analyse multi échelle sans la pertinence et points d'intérêt trouvés avec analyse multi échelle avec la pertinence des points.

2.3 Caractérisation des points d'intérêt

Comme nous venons de le voir, après avoir localisé les points d'intérêt et avoir mesuré leurs pertinences, nous allons les caractériser grâce aux invariants différentiels couleur de Hilbert [5,7].

Les invariants différentiels couleur donnent une description locale de l'image. Et comme le détecteur de Harris [4], ils ont la propriété d'être robustes aux principales transformations affines que peut subir l'image.

La description couleur est seulement basée sur les invariants du premier ordre de chaque composante couleur.

Il en résulte que chaque point d'intérêt *x* est décrit par un vecteur de huit invariants appelé $\vec{v}_{col}$.

$$\vec{v}_{col}\ (\vec{x},\ \sigma) = \begin{vmatrix} R \\ \|\nabla R\|^2 \\ G \\ \|\nabla G\|^2 \\ B \\ \|\nabla B\|^2 \\ \nabla R.\nabla G \\ \nabla R.\nabla B \end{vmatrix}$$

Cette caractérisation couleur locale semble être la méthode la plus pertinente pour décrire les points d'intérêt d'une image, car elle diminue significativement les principaux inconvénients des invariants différentiels [5,7] pour les images en niveaux de gris, du fait que nous n'utilisons que les dérivées d'ordre un, rendant ainsi le descripteur couleur relativement robuste vis à vis du bruit. De plus cette méthode donne la représentation la plus compacte possible, car elle résume avec par ses huit composantes toutes les informations photométriques couleur du voisinage du point d'intérêt. Cette caractérisation devient particulièrement intéressante pour des applications de recherche d'images (indexation) où nous allons comparer des points d'intérêt entre des images.

CHAPITRE 3:
SEGMENTATION DE L'IMAGE

Dans la partie précédente, nous avons traité de la façon de localiser et de caractériser les points d'intérêt d'une image couleur, cela dans le but de faire de l'indexation d'images. Le but de l'indexation d'images est de classer une série d'images par ordre de ressemblance par rapport à une image de référence, d'où l'utilisation des points d'intérêt, qui vont servir à définir la similitude entre les images. Pour calculer le degré de ressemblance entre deux images, nous allons prendre, un part un, les points d'intérêt d'une image que nous allons comparer avec tous les points d'intérêt de l'autre image.

Cependant nous nous apercevons rapidement que cette méthode peut prendre énormément de temps, si nous avons beaucoup de points d'intérêt. C'est pour cette raison que dans cette partie, nous présentons un moyen d'améliorer la mise en correspondance d'images.

La méthode que nous avons choisie est de diviser la mise en correspondance en deux étapes. Dans une première étape nous allons réaliser un appariement de graphe, puis dans un second temps faire la mise en correspondance par les points d'intérêt et d'autres critères entre les nœuds des graphes qui auront été mis en correspondance par l'appariement de graphes.

L'appariement de graphe revient à faire une mise en correspondance, mais au lieu de comparer tous les points d'intérêt des images, nous allons comparer leur graphe (ou leur segmentation). Donc si nous voulons pouvoir améliorer la mise en correspondance, il faut avoir la segmentation la plus fine possible, c'est à dire ayant le moins de régions possible. Car dans un graphe, les nœuds représentent les régions de l'image (un nœud par région) et les arcs représentent le fait qu'il y ait ou pas adjacences entre les régions (si il y a un arc entre deux nœuds, cela veut dire que les deux régions, correspondant aux nœuds, se touchent).

Dans cette partie, nous avons étudié la façon de simplifier au mieux les segmentations des

images, c'est à dire de faire de la décimation de graphe. La méthode est de réaliser dans un premier temps une sur-segmentation de l'image, puis de trouver un opérateur qui regroupera correctement toutes les régions qui sont proches au sens d'une distance. Une fois que l'appariement de graphes sera fait, il ne restera plus qu'à comparer tous les points d'intérêt d'une région d'une image avec ceux d'une région identique dans l'autre image. Cela nous donnera le degré de similitude entre les images, et permettra donc d'indexer une image dans une base de données.

Ainsi grâce à cet opérateur, nous allons simplifier la mise en correspondance par points d'intérêt.

Il existe plusieurs méthodes de décimation de graphes, mais dans notre étude nous n'avons traité que la méthode des cocons [3,6].

Pour réaliser la méthode des cocons, il nous faut réaliser dans un premier temps une sur-segmentation des images. Pour réaliser cette sur-segmentation des images, nous avons d'abord filtré les images grâce à la diffusion couleur puis nous avons calculé le gradient des images filtrées. Enfin pour trouver une sur-segmentation, nous avons utilisé la ligne de partage des eaux (LPE) à partir des minima locaux de la dérivée (cf. Figure 4).

Figure 4 : Exemple de sur-segmentation d'une image. Avec l'image initiale, l'image filtrée grâce à la diffusion couleur, le gradient de l'image filtrée, les minima locaux et la sur-segmentation de l'image.

Il existe plusieurs distances permettant de caractériser la similitude entre deux régions. Dans le paragraphe suivant, nous allons en passer quelques unes en revue.

3.1 Les différentes distances utilisées pour caractériser la similitude

3.1.1 La distance Euclidienne

$$d_{Euclidienne} = \sqrt{(\overline{R_1} - \overline{R_2})^2 + (\overline{G_1} - \overline{G_2})^2 + (\overline{B_1} - \overline{B_2})^2}$$

Dans cette formule, nous remarquons que la similarité ne prend en compte que la différence des moyennes couleur de chaque région.

Pour améliorer le calcul de similarité entre les régions, nous pouvons améliorer le calcul en y rajoutant les variances des régions.

3.1.2 La distance de Fisher

Une première façon, que nous avons utilisée pour rajouter la variance dans le calcul de la similarité entre les régions, est d'utiliser la distance de Fisher.

$$d_{Fisher} = \frac{(Nb\_pixel_1 + Nb\_pixel_2)}{(Nb\_pixel_1 \times \mathrm{var}_1) + (Nb\_pixel_2 \times \mathrm{var}_2)} \times d^2_{Euclidienne}$$

avec :

$$\mathrm{var}_i = \left| \frac{\mathrm{var}\_R_i + \mathrm{var}\_G_i + \mathrm{var}\_B_i}{3} \right| \qquad \text{et} \qquad \mathrm{var}\_R_i = \frac{R_i^2}{Nb\_pixel_i} - \frac{R_i^2}{(Nb\_pixel_i)^2}$$

avec $R_i = \sum$ des composantes du vecteur couleur R de chaque pixels de la région i ;

idem pour les deux autres vecteurs couleur (G et B).

3.1.3 La distance de Student

La deuxième méthode que nous avons utilisée est la distance de Student.

$$d_{Student} = \frac{d^2_{Euclidienne}}{\sqrt{\frac{(Nb\_pixel_1 \times \mathrm{var}_1) + (Nb\_pixel_2 \times \mathrm{var}_2)}{(Nb\_pixel_1 + Nb\_pixel_2 - 2)}} \times \sqrt{\frac{1}{Nb\_pixel_1} + \frac{1}{Nb\_pixel_2}}}$$

3.2 La décimation de graphe par la méthode des cocons

3.2.1 Introduction

Cette partie décrit une première approche de la segmentation qui s'appuie sur le formalisme de la théorie des graphes et aboutit à des ensembles, appelés *cocons*, qui s'organisent en hiérarchies. Le modèle de cocon [3,6] a initialement été motivé par la remarque de certains défauts dans les formulations « classiques » de la segmentation d'image par prédicats d'homogénéité.

Le point de départ du modèle de cocon provient d'une interrogation sur la manière d'exprimer la pertinence visuelle d'une région individuelle d'image. Les cocons tentent de formaliser une intuition courante : un certain nombre d'objets visuellement pertinents se caractérisent par le fait qu'ils se distinguent de leur environnement du point de vue d'une qualité donnée (intensité lumineuse, couleur, texture…). Par exemple, une orange posée sur une table, même si l'éclairage provoque des dégradés de lumière sur sa surface bombée, se distingue de son environnement par sa couleur caractéristique. Ou encore une feuille de papier qui serait posée sur la table à côté de l'orange apparaîtrait globalement plus claire que son environnement.
Nous avons alors cherché à caractériser ces zones remarquables par des propriétés ne dépendant d'aucun paramètre extérieur mais uniquement de caractéristiques intrinsèques à l'image. Pour fixer une terminologie, disons d'une région qu'elle est contrastée quand elle possède la propriété de se distinguer de son environnement pour une caractéristique visuelle donnée.

3.2.2 Définition

La notion de cocon [3,6] est définie sur un triplet (X, U, d), où G = (X, U) est un graphe non orienté et d'une mesure de dissemblance portée par les arêtes de G.
X représente les objets élémentaires que nous cherchons à regrouper en ensembles « pertinents » (dans notre étude, ce sont les régions de l'image).
U est un ensemble d'arêtes qui définissent une topologie sur X (relation de voisinage).
d est une fonction de U, dont le rôle est de qualifier le degré de différence accordé aux objets voisins dans G (dans notre étude cela représente l'éloignement entre les régions de l'image).

Dans notre étude, les graphes seront des graphes d'adjacence de régions construits à partir

d'une segmentation fine de l'image. Avec ces hypothèses, nous appellerons cocycle d'un sous graphe induit Y, l'ensemble des arêtes traversant sa frontière (c'est à dire qu'une extrémité se trouve dans Y et l'autre extrémité dans X\Y), et arête interne, l'ensemble des arêtes se trouvant entièrement comprises dans le sous graphe induit Y (cf. Figure 5).

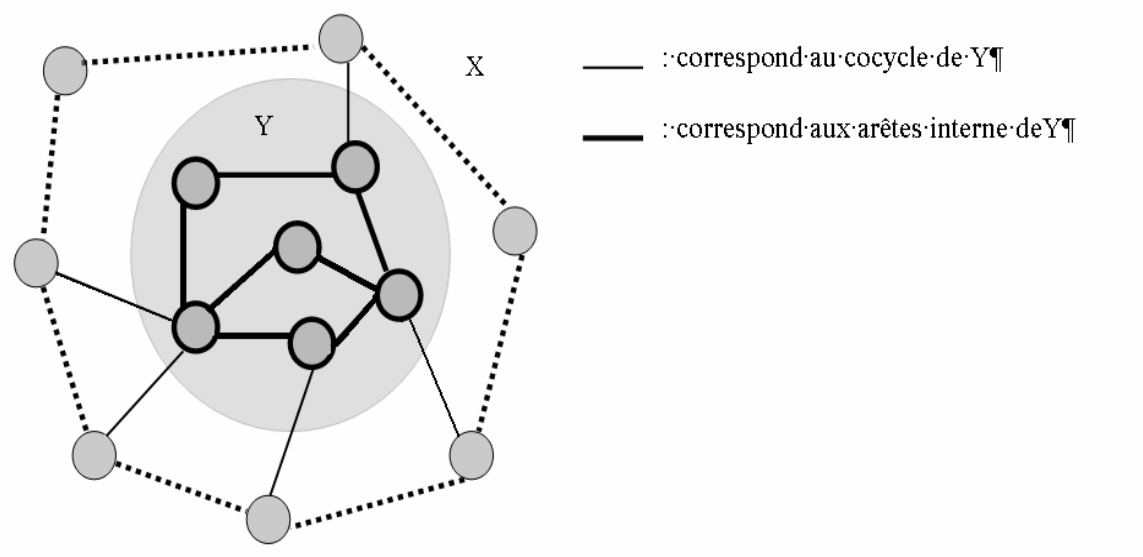


Figure 5 : Illustration du cocycle d'un sous graphe et de ses arêtes externes.

3.3 La hiérarchie des cocons durs (max-min)

Nous ne nous sommes intéressés, dans cette étude, qu'à une famille de cocons appelés cocons durs [3,6].

Dans ces conditions, la construction d'un pas d'agglomération des cocons se fait de la manière suivante :

-> Pour un sous graphe induit donné (les régions les plus proches), nous calculons son contraste interne. Dans les cocons durs, le contraste interne est donné par la valeur maximale des arêtes internes.

-> Puis nous calculons le contraste externe, donné par la valeur minimale du cocycle du sous

graphe induit.

->Enfin le sous graphe induit est un cocon si et seulement si le contraste interne est strictement inférieur au contraste externe.

Pour obtenir le cocon maximal [3,6], il suffit d'itérer cette construction jusqu'à ne plus pouvoir faire d'agglomérations ou de tomber sur une région appartenant déjà à un autre cocon, et ceci pour toutes les régions du graphe.

Les cocons durs modélisent des ensembles dont la plus forte dissemblance interne est strictement plus faible que la plus petite dissemblance externe. Ainsi, si la dissemblance traduit les variations locales d'une image, alors les cocons durs sont les groupes connexes de régions tels que la plus forte variation interne au groupe est plus faible que la plus petite variation sur la frontière du groupe. Autrement dit, un cocon dur possède une frontière extérieure nette, partout plus nette que n'importe laquelle de ses frontières internes.

Les cocons durs peuvent strictement s'envisager comme des portions d'image « lisses » localement maximales, tels que tout élargissement les rend moins lisses.

Cependant l'algorithme des cocons durs ne nous donne pas une segmentation assez simplifiée du fait qu'il s'agit également d'un algorithme de sur-segmentation. C'est pour cela que nous avons décidé de créer un programme qui itérerait l'algorithme des cocons. Mais il faut définir un critère d'arrêt de ce processus itératif.

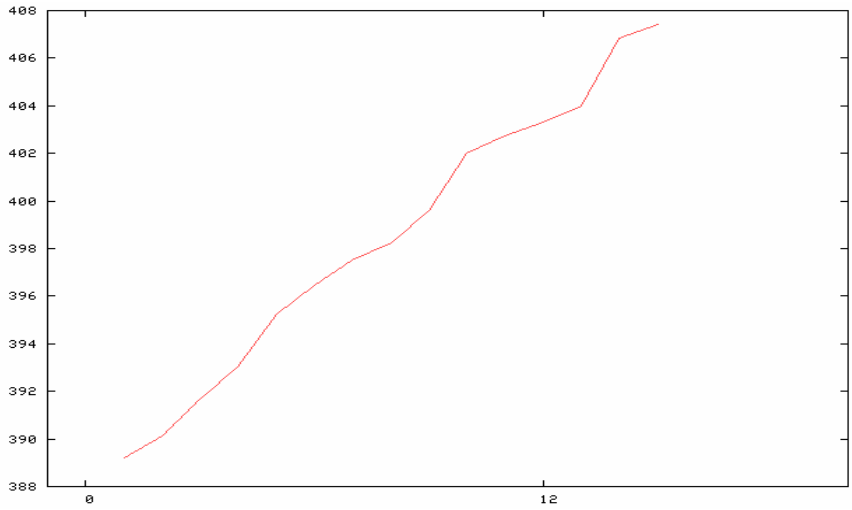
Pour faire cela, après chaque exécution de l'algorithme des cocons [3,6], nous mesurons l'erreur entre l'image initiale et l'image obtenue par reconstruction de la carte segmentation créée par l'algorithme des cocons (cf. Figure 6). Puis nous déterminons la dérivée de l'erreur en fonction du nombre d'itérations (cf. Figure 7). Enfin le processus d'itération de l'algorithme des cocons s'arrête lorsque la dérivée de l'erreur est minimale.

Pour déterminer l'erreur entre l'image initiale et l'image obtenue, nous avons calculé la moyenne de l'erreur au carré (MSE) entre les pixels des deux images.

$$MSE = \frac{\sum_{i=1}^{i=L}\sum_{j=1}^{j=N}(im1\_R[i][j]-im2\_R[i][j])^2+(im1\_G[i][j]-im2\_G[i][j])^2+(im1\_B[i][j]-im2\_B[i][j])^2}{3\times N\times L}$$

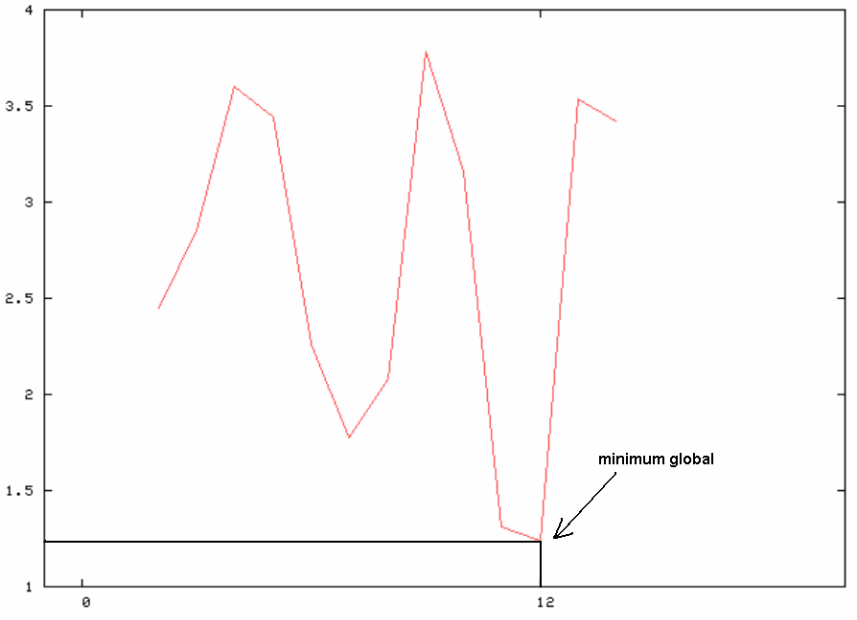
avec : L le nombre de lignes et N le nombre de colonnes.

Pour calculer la dérivée de l'erreur MSE en fonction du nombre d'itérations, nous avons réalisé 20 fois l'algorithme des cocons [3,6] (sauf s'il n'y a plus de fusion de régions possible), en gardant en mémoire les résultats obtenus après chaque exécution. Puis nous avons calculé la dérivé de l'erreur MSE avec un filtre [-1 0 1], c'est-à-dire que pour chaque exécution i de l'algorithme des cocons la dérivée vaut : $\nabla MSE_i = MSE_{i+1} - MSE_{i-1}$. Après avoir calculé la dérivée, nous avons recherché le minimum global. Si nous trouvons un minimum global l'algorithme s'arrête, sinon nous réalisons à nouveau 20 fois l'algorithme des cocons, nous recalculons la dérivée et nous cherchons à nouveau le minimum global en ayant mémorisé la plus petite valeur de la dérivée trouvée lors de la recherche précédente. Et ceci ainsi de suite jusqu'à trouver un minimum global.



Nombre d'itérations de l'algorithme des cocons.

Figure 6 : Exemple de courbe d'erreur entre image de départ et image reconstituée.



Nombre d'itérations de l'algorithme des cocons.

Figure 7 : Exemple de dérivée de la courbe d'erreur.

De plus nous avons défini, dans le paragraphe **3.1**, différentes distances permettant de calculer la similarité entre les régions. Nous avons pu remarquer, après avoir réalisé l'algorithme de segmentation avec toutes ces distances, que c'est la distance de Student qui nous permet d'avoir les meilleurs résultats (cf. Figures 8a, 8b et 8c).

Figure 8a : Image initiale, segmentation fine de l'image avec la distance de Student et image reconstruite à partir de la segmentation fine.

Figure 8b : Image initiale, segmentation fine de l'image avec la distance de Fisher et image reconstruite à partir de la segmentation fine.

Figure 8c : Image initiale, segmentation fine de l'image avec la distance Euclidienne et image reconstruite à partir de la segmentation fine.

Après avoir remarqué que c'est la distance de Student qui nous donne les meilleurs résultats, nous n'avons gardé dans notre algorithme que la distance de Student pour réaliser les segmentations fines des images.

De plus afin de simplifier davantage la segmentation, nous avons fusionné toutes les régions de la carte qui étaient « petites » avec la région qui leur est la plus proche au sens d'une distance. Pour déterminer la taille d'une région qui serait considérée comme « petite », nous avons fait l'histogramme des tailles des régions sur plusieurs images (cf. Figure 9).

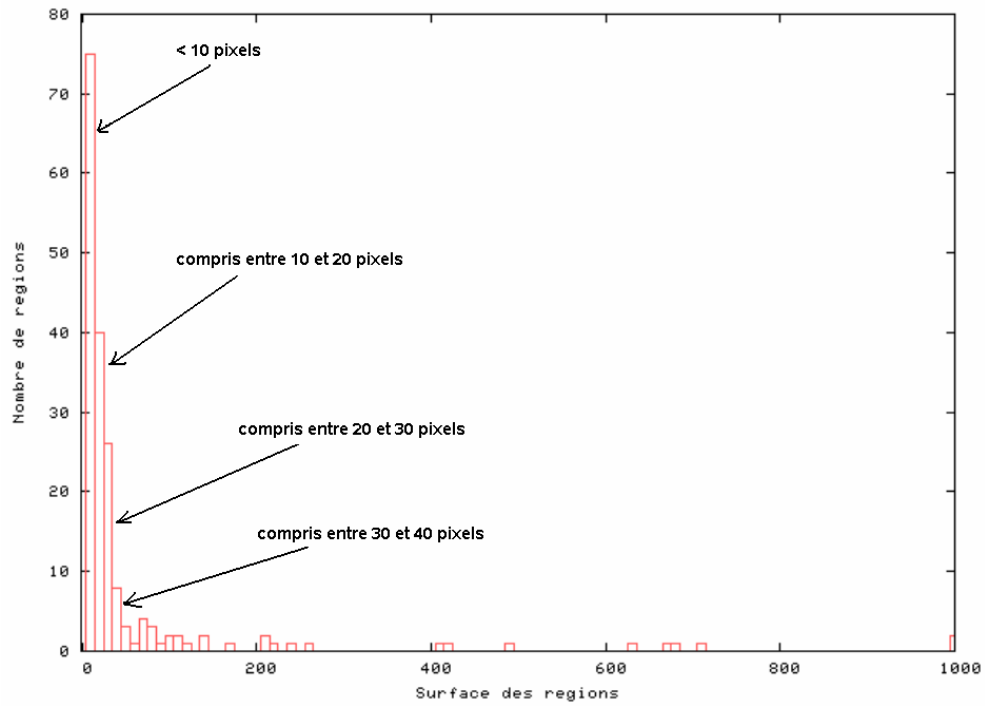


Figure 9 : Image sur-segmentée et segmentation fine de l'image.

Nous pouvons remarquer sur l'histogramme (cf. Figure 9), qu'il y a énormément de régions ayant une surface inférieure à 30 pixels comparé au nombre de régions ayant une surface comprise dans les autres intervalles de taille. Afin de simplifier d'avantage la segmentation, nous allons fusionner toutes ces régions ayant une surface inférieure à 30 pixels.

Grâce à tous ces critères, nous avons réussi à implémenter un algorithme permettant de réaliser une segmentation fine des images.

Les Figures 10a et 10b illustrent l'évolution de la segmentation de l'image au fur et à mesure des itérations de l'algorithme des cocons [3,6]. Dans cet exemple (cf. Figures 10a et 10b), le processus itératif de l'algorithme des cocons s'arrête, après quatre itérations, car il n'y a plus de fusion possible.

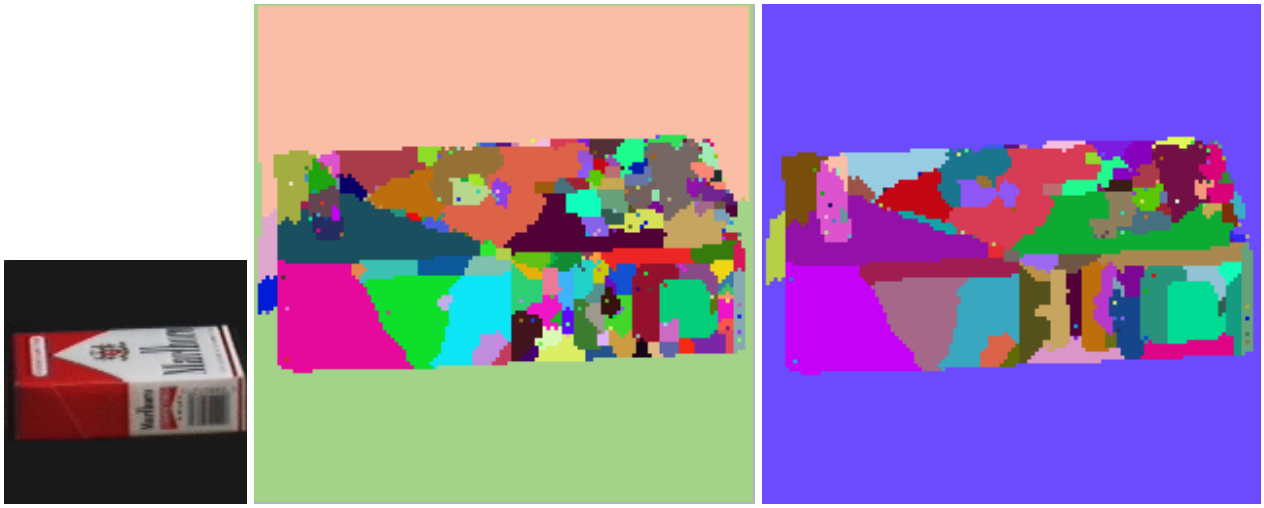


Figure 10a : Image initiale et image sur-segmentée et image après deux exécutions de l'algorithme des cocons.

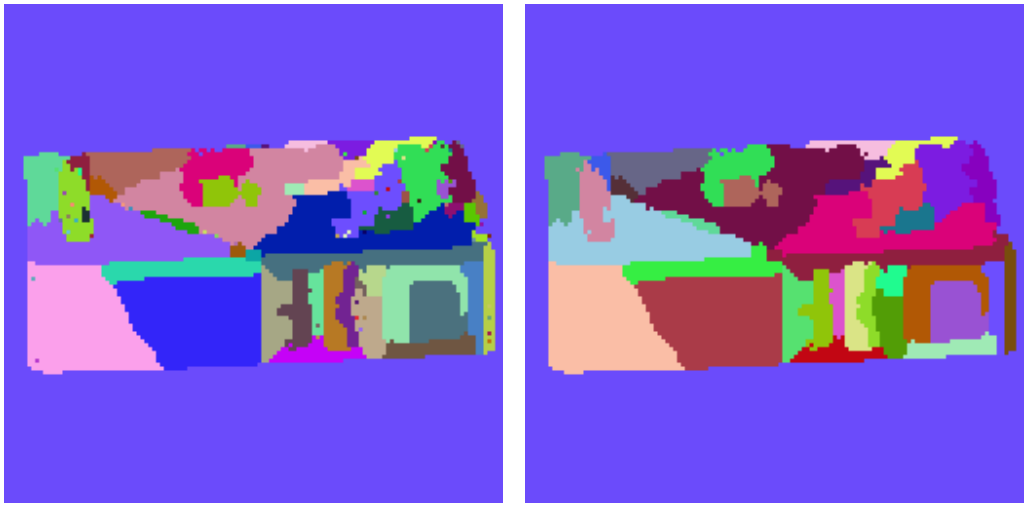


Figure 10b : Image après quatre exécutions de l'algorithme des cocons et image après fusion des régions « petites » avec la région qui leur est la plus proche.

Maintenant, après avoir défini les méthodes qui vont servir à réaliser la mise en correspondance, nous allons passer à la partie concernant l'indexation d'images proprement dite.

CHAPITRE 4:
MISE EN CORRESPONDANCE D'IMAGES

Nous venons de voir, dans les chapitres précédents, les différents principes qui vont nous servir pour réaliser la mise en correspondance de deux images. Dans ce chapitre, nous allons détailler comment nous avons réalisé la mise en correspondance d'images. Pour expliquer ceci, nous allons étudier la mise en correspondance de graphe, et la mise en correspondance des points d'intérêt, qui sont les deux parties du programme de mise en correspondance d'images.

Nous commencerons tout d'abord par faire un rapide état de l'art sur les principales approches de mise en correspondance de graphe. Puis nous traiterons de la façon dont nous avons, dans notre projet, réalisé la mise en correspondance de deux images.

Enfin nous expliquerons notre méthode pour effectuer l'indexation d'images. L'indexation d'images n'est qu'un processus récursif de mise en correspondance d'une image (que nous appellerons « image requête ») avec toutes les images d'une base, dans laquelle nous souhaitons indexer notre « image requête ».

4.1 Etat de l'art

4.1.1 Introduction

L'objectif de toute technique de mise en correspondance de graphes est d'établir un appariement parmi l'ensemble des nœuds des deux structures. Lorsque ces algorithmes s'appliquent aux graphes de voisinage, pour lesquels chacun des nœuds représente une région de la partition, ceci devient un processus de mise en correspondance de partitions. La Figure 11 illustre les données présentes à l'entrée et à la sortie d'un algorithme de mise en correspondance de graphes.

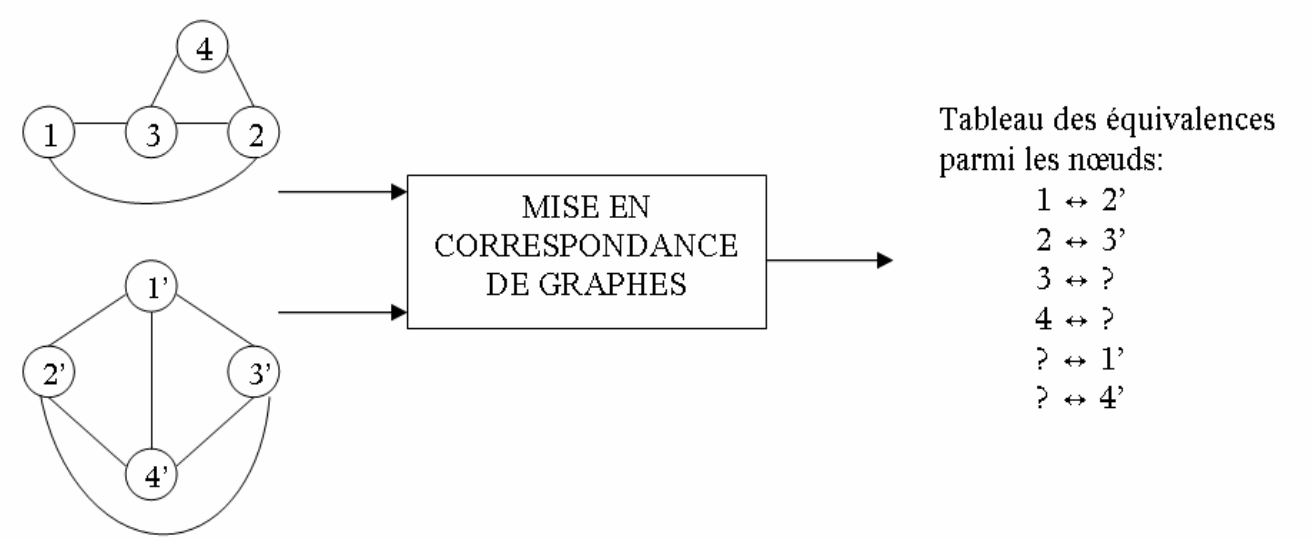


Figure 11 : Illustration des données en entrée et en sortie d'un algorithme de mise en correspondance.

On retrouve à l'entrée les graphes de voisinages obtenus à partir de deux partitions, et à la sortie un tableau d'équivalences parmi leurs ensembles de nœuds. Notons que s'il n'existe pas une équivalence parfaite parmi les structures des graphes, il se peut que certains des nœuds ne trouvent pas de correspondant (symbolisé par « ? » dans le tableau). A l'origine, il suffit qu'un objet ait été segmenté en un nombre différent de régions (le nœud 3 dans A apparaît fragmenté dans B), ou tout simplement qu'un objet ne soit pas présent dans les deux partitions (le nœud 4 du graphe A n'apparaît pas dans B).

Dans un cadre plus général, où le but ultime de l'algorithme est d'évaluer la ressemblance parmi les deux graphes à l'entrée, l'algorithme de mise en correspondance procéderait aussi au calcul d'une fonction de coût pondérant la dissimilarité des structures.

4.1.2 Les différentes approches

Il existe une abondante littérature concernant la théorie des graphes et de mise en correspondance de graphes. Notre but ici est de donner une vision globale des principales approches dans ce domaine.

Depuis longtemps, les graphes s'appliquent avec succès au problème de la reconnaissance

d'objets, comme par exemple, la reconnaissance de caractères manuscrits (Lue et al. [8]), de diagrammes (Messmer et Bunke [9]) ou même de formes dans un sens plus large (Siddiqi et al. [10]). Pour la plupart de ces applications, les graphes servent à modeler les objets connus à priori, ceux-ci étant ensuite stockés dans de larges bases de données. Le problème de l'identification d'un élément inconnu se transforme ainsi en un problème de mise en correspondance de graphes.

Les premières approches de mise en correspondance de graphes cherchent à établir un appariement parfait parmi les graphes, connu comme isomorphisme de graphes (Read et Corneil [11]). Un isomorphisme de graphes est une projection bijective parmi les nœuds de deux graphes. Cette définition impose aux graphes d'avoir le même nombre de nœuds, les mêmes labels, et une structure d'arêtes identiques.

Toutefois, on retrouve rarement une correspondance parfaite parmi les graphes modèles et ceux qui sont obtenus à partir des données en provenance d'une prise réelle, soit à cause du bruit dans le processus d'acquisition, soit à cause des erreurs d'interprétation lors de l'abstraction de la structure du graphe.

Pour surmonter ces inconvénients, il a été nécessaire de concevoir des techniques de mise en correspondance approximatives, dites aussi inexactes. Parmi celles-ci on peut différencier clairement trois types d'approches :

1) Mise en correspondance par isomorphisme restreint à des sous-graphes :

Ces techniques cherchent à identifier des sous-structures identiques à l'intérieur de structures plus larges (Ullman [12], Shapiro [13]). Le graphe d'association est un graphe auxiliaire dont chaque nœud est créé par appariement de deux nœuds des graphes d'entrés. Notons qu'il est possible d'imposer des contraintes pour limiter le nombre d'appariements et ainsi la taille du graphe. Deux nœuds sont connectés sur le graphe d'association, si et seulement si leurs topologies sur les graphes d'origine sont compatibles. Tout sous-ensemble de nœuds mutuellement connectés dans le graphe d'association est connu comme *clique*. Sous cette approche, la mise en correspondance des graphes revient à la recherche de la *clique maximale*, c'est à dire celle qui contient le plus de nœuds.

2) Mise en correspondance par édition des graphes :

Dans cette catégorie nous avons regroupé toutes les techniques qui incorporent des opérations d'édition au processus de mise en correspondance. Les opérations d'édition permettent de relabelliser, effacer ou insérer des nœuds ainsi que des arêtes jusqu'à ce que l'isomorphisme de graphes soit atteint. Cette idée a été originalement proposée par Sanfeliu et Fu dans [14] et fortement développée par Bunke dans de nombreux articles [15,16,9,17].Pour toutes ces approches, il s'agit de calculer une distance parmi les graphes en fonction du nombre d'éditions nécessaires pour obtenir des structures identiques. A chaque classe d'opération nous devrons lui associer un coût d'édition, de façon à ce que la mise en correspondance de graphes devienne la minimisation de cette fonction de coût.

3) Mise en correspondance par inclusion de nœuds fictifs :

Finalement, faisant partie des techniques de mise en correspondance approximatives, on doit classer dans une catégorie à part les méthodes qui traitent les différences parmi les graphes en introduisant une nouvelle classe de nœuds nommés nœuds fictifs. L'idée ici est d'apparier les nœuds qui causent la différence parmi les structures vers des nœuds fictifs. Lors de la mise en correspondance nous allons inclure autant de nœuds fictifs que nécessaire pour assurer un appariement consistant des voisinages. Shapiro et Haralick [13,18] utilisaient cette stratégie pour la première fois sans pénaliser l'inclusion de nœuds fictifs, d'autres approches comme celle adoptée par Boyer et Kak dans [19] ainsi que celle de Christmas et al. Dans [20] prévoient un coût d'insertion.

Le problème de la mise en correspondance étant très complexe, il n'est pas possible de présenter une seule de ces techniques comme la plus performante. Tout simplement, le choix de telle ou telle stratégie va être lié au type d'application visée.

Mais quelle que soit la démarche choisie, nous constatons une croissance exponentielle de la complexité de calcul en fonction de la taille des graphes sous analyse. De plus la plupart des méthodes appliquées à des situations réelles ont dû incorporer de l'heuristique afin de simplifier le problème et d'augmenter la robustesse des appariements.

Nous conclurons ainsi la revue introductive des techniques concernant la mise en correspondance de graphes. Par la suite nous préciserons le choix fait pour l'algorithme de mise en correspondance de graphes. Pour une étude plus approfondie des différentes méthodes définie ci-dessus, nous pouvons nous reporter à [1].

4.2 Mise en correspondance d'images

Après avoir fait un rapide état de l'art sur la mise en correspondance de graphe, nous allons détailler dans cette partie l'algorithme que nous avons utilisé pour réaliser la mise en correspondance de graphe. Puis dans un second temps, nous traiterons de la mise en correspondance des points d'intérêt. Enfin dans la dernière partie, nous décrirons notre programme complet de mise en correspondance, et notre programme d'indexation d'images, qui n'est comme nous l'avons déjà dit qu'une itération du programme de mise en correspondance sur toute une base d'images.

4.2.1 Mise en correspondance de graphe

Première approche :

Pour réaliser la mise en correspondance de graphe, nous sommes partis dans un premier temps sur la méthode introduite par la thèse de [1]. Dans cette méthode, la mise en correspondance de graphe est réalisée en 3 étapes (cf. Figure 12).

● La première étape, appelée édition par resegmentation des régions, est une étape de resegmentation des partitions d'entrées, cette étape permet d'essayer de faire se ressembler le plus possible les deux partitions. Le but de cette étape est de palier le problème de segmentation car deux partitions identiques peuvent être segmentées différemment, par exemple s'il y a eu un changement d'illumination. Le but de l'édition est donc de resegmenter les partitions de façon à ce que toute région en commun sur les images de départ soit également découpée.

● La deuxième étape, qui est la mise en correspondance de graphe, est réalisée grâce à un algorithme

de type inexact à isomorphisme par inclusion de nœuds fictifs. De plus, afin de simplifier le problème et d'améliorer la robustesse des appariements, [1] a intégrée une heuristique (relaxation probabiliste) dans son algorithme.

● La troisième étape, appelée édition par fusion des régions, va conclure le processus de mise en correspondance en établissant un chemin d'étiquetage qui relie les deux partitions de départ. Le but de l'algorithme de fusion est de délivrer comme résultat de la mise en correspondance la partition la plus proche à celle qui joue le rôle de référence, tout en restant fidèle au découpage fourni par la segmentation multi échelle.

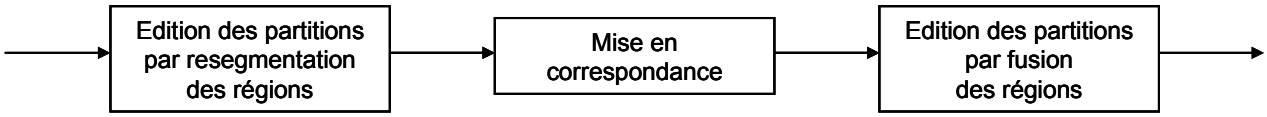


Figure 12 : Mise en correspondance de graphes.

Cependant pour notre problème cet algorithme est difficile à mettre en place, car il faut garder en mémoire, pour l'édition par resegmentation des régions, toutes les fusions qui sont réalisées depuis la sur-segmentation jusqu'à la segmentation fine de l'image. Ce qui prendrait énormément de place mémoire sachant qu'à chaque itération de l'algorithme des cocons [3,6] nous réalisons plusieurs fusions, et que nous itérons l'algorithme des cocons plusieurs fois.

C'est pour cette raison que nous sommes passés à un autre algorithme de mise en correspondance de graphe.

Deuxième approche :

Dans cette approche, nous avons implémenté un algorithme de type inexact à isomorphisme restreint à des sous graphes. Comme nous l'avons vu dans le paragraphe **4.1**, ce type d'algorithme revient à la recherche de la *clique maximale*, c'est-à-dire celle qui contient le plus de nœuds.

Pour réaliser cet algorithme, nous l'avons implémenté, comme il est défini dans [2], sous la forme d'un arbre (cf. Figure 13). A chaque niveau de l'arbre, un nœud représente un appariement possible entre une région d'une image et une région de l'autre image. Pour avoir un algorithme

rapide, nous ne traitons que les couples de régions dont la dissimilarité entre les deux régions composant le nœud est inférieur à un seuil τ. Un arc est ajouté à un chemin de l'arbre seulement s'il y a cohérence topologique entre les deux paires de région. Un nœud est développé au niveau inférieur si et seulement si la dissimilarité totale du chemin est inférieure à un seuil T, la dissimilarité totale du chemin étant calculée en additionnant les dissimilarités des nœuds composant le chemin. Par exemple dans la Figure 13, le chemin {(R1,S2) (R2,S2) (R4,S7)} a une dissimilarité totale suffisamment faible, alors que le chemin {(R1,S2) (R2,S2) (R4,S6)} a une dissimilarité totale trop importante.

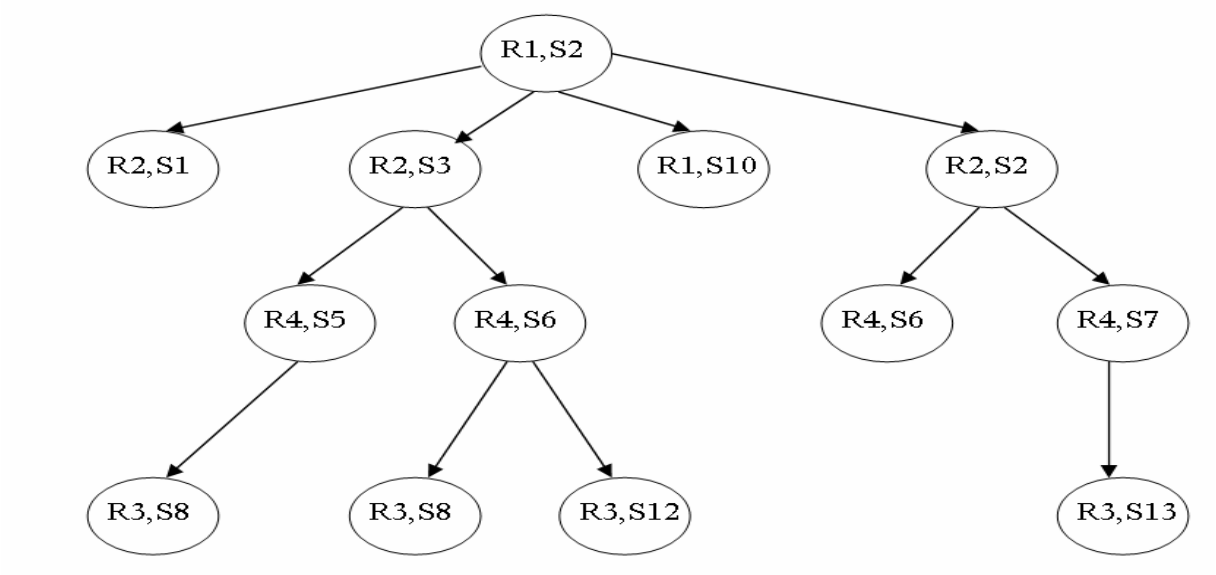


Figure 13 : Arbre de recherche de l'algorithme de mise en correspondance.

Nous avons parlé, un peu plus haut, de cohérence topologique. Dans notre étude la cohérence topologique est définie par deux critères. Il y a cohérence topologique entre les couples (Ri,Si) et (Ri+1,Si+1) si :

- l'adjacence de (Ri,Ri+1) et (Si,Si+1) est du même type (adjacent ou non).
- la position relative de (Ri,Ri+1) et (Si,Si+1) est du même type, par exemple si Ri+1 est au dessus de Ri alors Si+1 doit être au dessus de Si (dans notre étude, on ne prend en compte

si Ri+1 est à droite ou a gauche de Ri).

Pour déterminer si une région est au dessus ou en dessous d'une autre, il faut définir un point de référence pour chaque région. Un critère simple pour caractériser la localisation d'une région dans une image est le barycentre. Cependant les centres de gravité ne se trouvent pas nécessairement sur la région, et peuvent donc se superposer. La Figure 14 illustre le problème lié au centre de gravité. Afin d'éviter ce problème, nous avons utilisé comme point de référence le premier point de chaque région rencontré lors du balayage causal de l'image (c'est-à-dire le point de la région le plus haut dans l'image). Le balayage causal d'une image étant effectué en ligne à partir du coin haut gauche (cf. Figure 14).

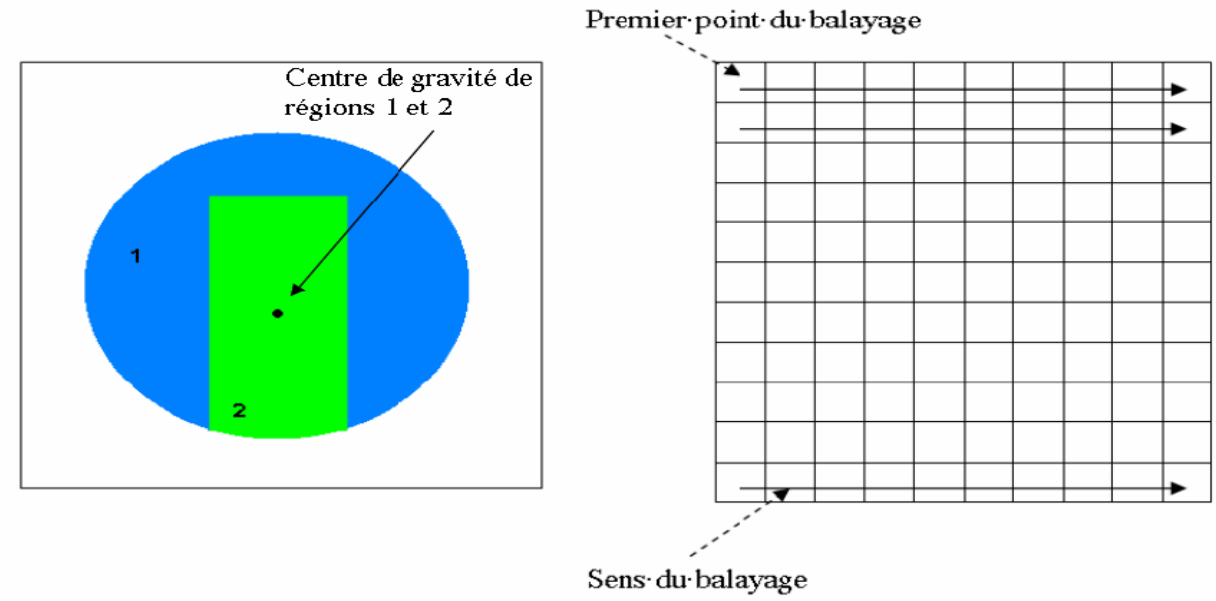


Figure 14 : Problème des centres de gravités, sens de balayage des images.

Un appariement entre deux sous graphes correspond à un chemin de la racine à une feuille de l'arbre de recherche. Comme la dissimilarité totale est une somme des dissimilarités des nœuds d'un chemin, il est intéressant de mettre aux niveaux les plus hauts de l'arbre (proche de la racine) les couples de régions très dissemblables, l'arbre sera ainsi plus rapidement élagué.

Pour construire l'arbre, nous calculons toutes les distances entre toutes les régions des deux images. Les couples pour lesquels la distance est supérieure à τ sont supprimés. Nous trions dans

une liste les couples restants par ordre décroissant (du moins semblable au plus semblable). Nous regardons dans la liste la région requête du premier couple (le moins semblable). Tous les couples de la liste ayant cette région requête seront placés au premier niveau de l'arbre. Nous cherchons ensuite, dans la liste, le premier couple ayant une région requête différente de celles des niveaux inférieurs. Tous les couples ayant cette région requête seront placés au second niveau de l'arbre, et ainsi de suite.

Les propriétés de cet arbre sont les suivantes (deux régions sont dites similaires si leur dissimilarité est inférieure à τ) :

- Une image qui n'a aucune région similaire avec une région de l'image requête est aussitôt écartée.
- La région requête qui correspond le moins avec une région de l'image cible est examinée en premier.
- Il y a au maximum autant de niveaux dans l'arbre que de région requête différente dans la liste. Il y a un niveau pour chaque région requête ayant au moins une région cible similaire.
- Dès que la dissimilarité totale entre les graphes dépasse une valeur donnée, le nœud en cours n'est plus développé.

Chaque chemin de la racine (premier niveau de l'arbre) à une feuille du dernier niveau représente un appariement de sous graphes acceptables. Par exemple dans la Figure 13, le chemin {(R1,S2) (R2,S3) (R4,S6) (R3,S12)} est l'un des quatre appariements de graphes possibles.

Nous remarquons qu'une région de l'image cible peut être appariée à plusieurs régions requêtes, mais pas l'inverse. On peut donc gérer une sous-segmentation de l'image cible par rapport à l'image requête, mais pas une sur-segmentation.

Après cette première étape de construction de l'arbre, une deuxième étape consiste donc à tenter d'apparier plusieurs régions cibles avec la même région requête. Pour cela, nous examinons les voisins de chaque région cible appariée et nous recherchons les régions susceptibles d'être appariées avec la région requête. Plus précisément, pour chaque appariement (Ri,Sj), nous regardons

tous les voisins Sk de Sj, si la distance entre Ri et Sk est inférieure à τ et que la dissimilarité totale est inférieure à T, alors Ri est mise en correspondance avec Sk.

Un appariement entre graphes est finalement un ensemble de couple région requête, région cible suffisamment semblables et dont les adjacences et les positions relatives sont les mêmes dans les deux sous graphes.

Pour le calcul de la dissimilarité totale, si nous appelons d(Ri,Sj) la dissimilarité des couples de régions de l'arbre, alors la dissimilarité totale d'un chemin est calculée de la façon suivante :

$d_{totale} = \frac{1}{m}\sum_{t=1}^{m} d(Ri, Sj)$ avec m le nombre de nœuds composant le chemin.

De plus, nous remarquons qu'il peut y avoir plusieurs appariements de graphes possibles (par exemple Figure 13, il y en a quatre), c'est-à-dire plusieurs chemins possibles donc plusieurs valeurs de dissimilarité totale. Ce que nous voulons à la sortie de l'algorithme, c'est une seule valeur de dissemblance entre les deux images d'entrées. La valeur finale de l'appariement entre deux images est déterminée de la façon suivante :

$D = \min\left\{d_{totale}^{\ i}\right\}$ où $d_{totale}^{\ i}$ représente la dissimilarité totale du chemin i.

Comme dans l'algorithme des cocons [3,6], nous calculons dans la mise en correspondance des dissimilarités entre des régions. Pour calculer ces dissimilarités entre régions, nous avons repris dans un premier temps les distances utilisées dans l'algorithme des cocons (distance Euclidienne, distance de Fisher et distance de Student).

Puis dans un second temps, nous avons essayé une nouvelle distance, tirée de la thèse de [1]. Cette distance se présente sous la forme d'une probabilité de mise en correspondance. Dans notre étude, nous n'avons repris qu'un des critères de similarité introduit dans la thèse de [1]. Le critère que nous avons utilisé porte sur la similarité couleur des régions (nous l'appellerons $\Gamma_{couleur}$).

Nous pouvons alors définir la probabilité de mise en correspondance (P) comme :

$P = \Gamma_{couleur}$

La similarité couleur des régions étant définie de la façon suivante (entre la région i et la région j) :

$$\Gamma_{couleur} = \left(1 - \frac{|R_i - R_j|}{R_{\max}}\right)\left(1 - \frac{|G_i - G_j|}{G_{\max}}\right)\left(1 - \frac{|B_i - B_j|}{B_{\max}}\right)$$

où R, G et B sont respectivement les valeurs moyenne des composantes rouge, vert et bleu des régions.

De plus $R_{\max} = G_{\max} = B_{\max} = 255$.

Avec cette distance, nous remarquons que deux régions proches auront une probabilité P tendant vers 1 alors que deux régions très différentes auront une probabilité voisine de 0. Ce qui ne va pas avec la construction de l'arbre que nous avons vu précédemment, car dans cette construction nous avons défini les régions proches comme inférieures à un seuil. Ceci vient du fait que dans la construction de l'arbre nous avons parlé de dissemblance entre les noeuds, alors que la probabilité de mise en correspondance (P) calcule la similarité entre les nœuds. Pour résoudre ce problème, nous avons modifié la probabilité de la façon suivante :

$$P' = 1 - P = 1 - \Gamma_{couleur}$$

Enfin, nous avons dit dans la construction de l'arbre, que nous pouvions gérer une sous-segmentation de l'image cible par rapport à l'image requête, mais pas une sur-segmentation. Sachant que nous voulons laisser le choix de construire l'arbre partiellement (seulement la première étape) ou totalement, nous avons rajouté un test dans notre algorithme permettant de toujours prendre comme image requête celle avec le moins de régions.

4.2.2 Mise en correspondance des points d'intérêt

Pour la mise en correspondance des points d'intérêt, nous avons défini, comme pour la similarité couleur, une distance sous la forme d'une probabilité de similarité entre les points d'intérêt. Pour cela, nous avons calculé la probabilité de similarité entre tous les paramètres des points d'intérêt. Les différents paramètres liés aux points d'intérêt sont les différentes caractéristiques des invariants différentiels couleur [5,7] (couleur, dérivées,…), la pertinence des points d'intérêt et leur position dans l'image. La probabilité de similarité totale ($D_{\text{int érêt}}$) est le

produit des différentes similarités.

La similarité couleur entre le point i et le point j :

$$d_{couleur} = \left(1 - \frac{|R_i - R_j|}{255}\right)\left(1 - \frac{|G_i - G_j|}{255}\right)\left(1 - \frac{|B_i - B_j|}{255}\right)$$

La similarité des dérivées couleur entre le point i et le point j :

$$d'_{dérivées} = \left(1 - \frac{|\nabla R_i^2 - \nabla R_j^2|}{255^2}\right)\left(1 - \frac{|\nabla G_i^2 - \nabla G_j^2|}{255^2}\right)\left(1 - \frac{|\nabla B_i^2 - \nabla B_j^2|}{255^2}\right)$$

$$d''_{dérivées} = \left(1 - \frac{|\nabla R \nabla G_i - \nabla R \nabla G_j|}{255^2}\right)\left(1 - \frac{|\nabla R \nabla B_i - \nabla R \nabla B_j|}{255^2}\right)$$

La similarité de pertinence entre le point i et le point j :

$$d_{pertinence} = \left(1 - \frac{|\Lambda_i - \Lambda_j|}{255}\right)$$ où Λ_i et Λ_j représente la pertinence du point i et du point j.

La similarité de position entre le point i et le point j :

$$d_{position} = \left(1 - \left|\frac{X_i}{Width_i} - \frac{X_j}{Width_j}\right|\right)\left(1 - \left|\frac{Y_i}{Height_i} - \frac{Y_j}{Height_j}\right|\right)$$

où $Width_i$ et $Width_j$ représente le nombre de colonne de l'image liée au point i ou au point j.

Et $Height_i$ et $Height_j$ représente le nombre de ligne de l'image liée au point i ou au point j.

La normalisation des valeurs des positions par les dimensions de l'image permet de nous affranchir de la différence de taille qui pourrait exister entre les deux images.

La similarité totale est ainsi déterminée de la façon suivante :

$$D_{\text{int érêt}} = d_{couleur} \times d'_{dérivées} \times d''_{dérivées} \times d_{pertinence} \times d_{position}$$

Cependant nous remarquons le même problème que pour la probabilité de mise en correspondance, c'est-à-dire que cette distance est minimale pour des points d'intérêt très éloignés les uns des autres. Nous avons donc effectué la même transformation :

$$D'_{\text{int érêt}} = 1 - D_{\text{int érêt}} = 1 - \left(d_{couleur} \times d'_{dérivées} \times d''_{dérivées} \times d_{pertinence} \times d_{position} \right)$$

Pour réaliser la mise en correspondance des points d'intérêt, nous allons toujours prendre comme région de référence celle avec le moins de point. Pour chaque point d'intérêt de la région de référence, nous allons calculer sa dissemblance avec tous les points de l'autre région et ne garder que la dissemblance la plus faible. Après avoir trouvé la dissemblance la plus faible pour chaque point de la région de référence, nous faisons la moyenne des dissemblances.

Après avoir défini nos algorithmes de mise en correspondance de graphe et de mise en correspondance des points d'intérêt, nous allons présenter maintenant comment nous avons regroupé nos deux algorithmes pour réaliser l'algorithme de mise en correspondance d'images, grâce auquel nous implémenterons notre programme d'indexation.

4.2.3 Mise en correspondance d'images, indexation

Pour réaliser notre programme de mise en correspondance d'images, nous sommes partis de l'algorithme de mise en correspondance de graphes auquel nous avons ajouté, au moment des calculs de dissemblance entre les régions d'une image et les régions de l'autre image, notre algorithme de mise en correspondance des points d'intérêt. Nous obtenons ainsi deux valeurs de dissemblances entre les régions des deux images, l'une basée sur les points d'intérêt et l'autre sur les régions. Nous avons donc fait, dans le cas où nous construisons l'arbre avec la probabilité de mise en correspondance ($P' = 1 - \Gamma_{couleur}$), la multiplication de ces deux dissemblances pour obtenir la dissemblance globale.

Mais toutes les régions ne contiennent pas forcément des points d'intérêt. Il en résulte deux méthodes possibles pour implémenter notre algorithme, soit nous analysons toutes les régions, soit nous n'analysons que les régions contenant des points d'intérêt.

Si nous analysons toutes les régions, dans le cas où l'une des deux régions (ou les deux) n'a pas de points d'intérêt, alors la dissemblance totale est seulement égale à la dissemblance liée aux régions.

Mais si nous n'analysons que les régions ayant des points d'intérêt, alors la construction de

l'arbre est modifiée. Dans la construction de l'arbre, nous avions trié dans une liste tous les couples de régions inférieures à un seuil τ. Ici nous allons trier dans la liste tous les couples de dissemblance inférieure à τ, et dont les deux régions du couple possèdent des points d'intérêt. L'avantage de cette méthode est qu'elle est plus rapide, du fait que nous ne considérons que les régions comprenant des points d'intérêt.

Pour effectuer l'indexation d'images, nous avons d'abord créé notre base d'images avec toutes les caractéristiques nécessaires (segmentation fine, caractéristiques des points d'intérêt), grâce aux programmes que nous avons définis dans les chapitres 2 et 3. Puis nous avons implémenté un programme réalisant l'algorithme de mise en correspondance d'images en boucle, à partir d'une image de référence sur toute la base d'images, et triant les images de la base suivant leur dissemblance par rapport à l'image de référence (de la plus similaire à la moins similaire).

4.3 Résultats

Après avoir exécuté notre algorithme d'indexation sur une base d'images, nous remarquons que le programme ne trie pas toujours correctement toutes les images (cf. Annexe A). Pour comprendre d'où proviennent ces erreurs d'indexation, nous avons réalisé des tests avec les différents critères et pour les deux méthodes d'implémentation possibles de l'algorithme d'indexation (comparaison de toutes les régions ou seulement des régions comprenant des points d'intérêt). Pour réaliser ces tests, nous avons exécuté notre algorithme d'indexation en n'utilisant qu'un des critères, ceci pour tous les critères, puis pour toutes les combinaisons de critères possibles et ceci pour chacune des deux méthodes.

Après avoir réalisé tous ces tests, nous avons réussi à trouver d'où provenait l'erreur d'indexation. Nous pouvons l'expliquer sur un exemple. Pour la mise en correspondance de toutes les régions et en n'utilisant que le critère de similarité couleur (cf. Annexe B), nous nous apercevons

que nous avons une erreur d'indexation car pour l'image de la voiture rouge le programme nous ressort en premier, comme image la plus similaire, un bateau bleu. Cette erreur est causée par l'algorithme de mise en correspondance de graphe que nous avons implémenté [2]. Car pour construire l'arbre, comme nous l'avons déjà dit, nous ne gardons que les couples dont la dissimilarité est inférieure à un seuil τ, le programme s'arrête quand la dissimilarité totale du chemin est supérieure à un seuil T et la valeur que retourne le programme pour caractériser la similitude entre les deux images est la dissimilarité totale du chemin la plus faible. Nous remarquons que la valeur retournée par le programme est plus faible pour le bateau bleu que pour une autre voiture rouge. Ceci s'explique par le fait qu'il n'y a qu'un seul couple de région, entre la voiture rouge et le bateau bleu, qui est mit en correspondance, les autres étant supérieur au seuil τ. Mais ce couple a une valeur de dissimilarité très faible, c'est ce qui fait que le bateau est donné comme image la plus proche de la voiture. Pour résoudre ce problème, il faudrait soit trouver un autre moyen de tenir compte du nombre de nœuds, c'est-à-dire du nombre de régions apparié, composant le chemin, soit trouver un moyen de tenir compte du nombre de régions différentes entre les deux images, ou bien améliorer la mesure de dissimilarité couleur entre deux régions.

Conclusion

Notre objectif principal concernait la mise en correspondance d'images en vue de l'indexation.

Dans un premier temps, avant d'entreprendre la mise en correspondance des images, nous avons considéré comme prioritaire de mettre au point des outils de caractérisation des images couleur. Pour cela, nous avons présenté deux algorithmes permettant de caractériser les images couleur, l'un basé sur les cocons [3,6] réalise une segmentation fine des images, l'autre utilisant les points d'intérêt [4] ainsi que les invariants différentiels couleur [5,7] permet à la fois une détection et une description de ces points d'intérêt.

Dans un second temps, nous nous sommes intéressé au problème d'indexation d'images, cela revient à la mise en correspondance d'images. Nous avons présenté une nouvelle méthode qui combine des algorithmes de mise en correspondance de graphes avec des algorithmes de mise en correspondance par points d'intérêt. Nous nous sommes intéressé au large potentiel qu'une telle représentation de la scène peut offrir : d'un côté, la structure relationnelle du graphe fournit une plus grande robustesse au système ; de l'autre côté, les points d'intérêt permettent d'avoir une description locale de l'image.

Pour surmonter l'inconvénient principal attribué aux algorithmes de mise en correspondance par points d'intérêt (la durée du temps de calcul), nous avons orienté nos recherches dans le but de rajouter une mise en correspondance de graphes avant de réaliser le processus de mise en correspondance par points d'intérêt, et ainsi accélérer le temps de calcul des appariements. Ainsi, l'atout le plus important de notre méthode est de combiner un descripteur hiérarchique avec un descripteur local, ce qui permet d'avoir une bonne description de l'image, grâce à la description locale de l'image, mais en ayant un algorithme de mise en correspondance d'images rapide.

Finalement, nous pouvons envisager des travaux futurs concernant l'amélioration de algorithme de mise en correspondance, afin de réaliser une meilleure indexation, comme nous l'avons décrit dans notre remarque du paragraphe **4.3.**

Bibliographie

[1] Gomila, C. "Mise en correspondance de partitions en vue du suivi d'objets". Thèse de doctorat, Centre de Morphologie Mathématique, Ecole des Mines de Paris, 2001.

[2] Philipp-Folguet, S. et Lekkat, M. "Recherche d'images à partir d'une requête partielle utilisant la disposition des régions". Rapport de recherche, Equipes Traitement des Images et du signal, ETIS, octobre 2003.

[3] Guigues, L., Le Men, H. et Cocquerez, J.P. "The hierarchy of the cocoons of a graph and its application to image segmentation". *Pattern Recognition Letters*, 24 :1059-1066, 2003

[4] Da Rugna, J. et Konik, H. "Color interest points detector for visual information retrieval". *LIGIV*, Université Jean Monnet, St Etienne, France, 2002.

[5] Gouet, V. et Boujemaa, N. "Object-based queries using color points of interest". *IMEDIA Project*, INRIA, Rocquencourt, 2001.

[6] Guigues, L. "Modèles Multi-Echelles pour la segmentation d'Images". Thèse de doctorat, *Laboratoire MATIS*, ING, 2003.

[7] Gouet, V., Montesinos, P., Deriche, R. et Pelé, D. "Evaluation de détecteurs de points d'intérêt pour la couleur". RFIA, Paris, 2000.

[8] Lu, S.W., Ren, Y., et Suen, C.Y. "Hierarchical attributed graph representation and recognition of handwritten chinese characters". Pattern Recognition, 24 :617-632, 1991.

[9] Messmer, B. et Bunke, H. "Graphics Recognition : Lecture Notes in Computer Science", volume 1072, chapitre Automatic learning and recognition of graphical symbols in engineering drawing, pages 123-134. Springer, Berlin, 1996.

[10] Siddiqi, K., Shokoufandeh, A., Dickinson, S.J., et Zucker, S.W. "Shock graphs and shape matching". Dans Sixth International Conference on Computer Vision, Bombay, Inde, 1998.

[11] Read, R.C. et Corneil, D.G. "The graph isomorphism disease". J. Graph Theory, 1 :339-363, 1977.

[12] Ullman, J.R. "An algorithm for subgraph isomorphism". Journal of the Assotiation for

Computing Machinery, 23(1) :31-42, 1976.

[13] Shapiro, L.G. et Haralick, R.M. "Structural descriptions and inexact matching". IEEE Transactions on Pattern Recognition and Machine Intelligence, 3 :504-519, Septembre 1981.

[14] Sanfeliu, A. et Fu, K.S. "A distance measure between attributed relational graphs for pattern recognition". IEEE Transactions on Systems, Man, and Cybernetics, 13 :353-363, 1983.

[15] Bunke, H. et Allerman, G. "Inexact graph matching for structural pattern recognition". Pattern Recognition Letters, 1(4) :245-253, 1983.

[16] Bunke, H. "On a relation between graph edit distance and maximum commom subgraph". Pattern Recognition Letters, 18 :689-694, 1997.

[17] Bunke, H. "Error correcting graph matching : On the inuence of the underlying cost function". IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(9) :917-922, Septembre 1999.

[18] Shapiro, L.G. et Haralick, R.M. "A metric for comparing relational descriptions". IEEE Transactions on Pattern Analysis and Machine Intelligence, 7 :90-94, 1985.

[19] Boyer, K. et Kak, A. "Structural stereopsis for 3D vision". IEEE Transaction on Pattern Analysis and Machine Intelligence, 10 :144-166, 1988.

[20] Chirstmas, W., Kittler, J., et Petrou, M. "Structural matching in computer vision using probabilistic relaxation". IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8) :749-764, 1995.

[21] Schettini, R., Ciocca, G. et Zuffi, S. "Color in databases: Indexation and Similarity". In First International Conference on Colour in Graphics and Image Processing Computer Graphics, and Image Processing'2000 , Saint-Etienne, France , October 2000 .

[22] Boughorbel, S., Boujemaa, N. et Vertan, C. "Description de la répartition spatiale de la couleur pour l'Indexation d'Images". RFIA'2002, vol. 2 pp. 407-414

[23] Clouard, R., Elmoataz, A., Angot, F., Duret-Lutz, A. et Lezoray, O. "Manuel de référence". Rapport de recherche, GREYC équipe Image, août 2004.

[24] Clouard, R., Elmoataz, A., Angot, F., Duret-Lutz, A. et Lezoray, O. "Manuel de programmation". Rapport de recherche, GREYC équipe Image, août 2004.

[25] Clouard, R., Elmoataz, A., Angot, F., Duret-Lutz, A., Lezoray, O., Schüpp, S., Bougleux, S., Belhomme, P., Quesnel, L. et Fadili, J. "Index des opérateurs". Rapport de recherche, GREYC équipe Image, août 2004.

Annexe A : Résultat du programme d'indexation

Indexation d'images

Image requête

Image trié par ordre de ressemblance:

Indexation d'images

Image requête

Image trié par ordre de ressemblance:

Annexe B :

Mise en correspondance de toutes les régions avec le critère de similarité couleur seule :

Annexe C :

Mise en correspondance, de toutes les régions, avec tous les critères :

Annexe D :

Mise en correspondance, que des régions contenant des points d'intérêt, avec tous les critères :