Reconnaissance d'objets
Approche structurelle

Guillaume Ryder

Stage de MIM 1 encadré par N. Zlatoff et B. Tellez
30 mai – 27 juillet 2005

Version du 5 septembre 2005

Laboratoire d'InfoRmatique
en Images et Systèmes d'information

CNRS UMR 5205

Sommaire

Introduction

On se propose d'étudier la reconnaissance d'objets en insistant sur l'aspect structurel. Cette reconnaissance s'effectue dans le cadre d'un moteur de recherche : on dispose d'une base de données d'images sur laquelle l'utilisateur peut exécuter une requête en fournissant un *modèle*. Le moteur de recherche doit comparer les images au modèle fourni et retourner un classement par similarité décroissante.

FIG. 1 – Exemple de recherche. À gauche le modèle, à droite deux images de la base.

Pour comparer deux images, on distingue habituellement trois niveaux d'interprétation :

- **Bas niveau.** On reste à l'échelle des pixels et du signal pour comparer les images selon la couleur ou les textures, par exemple en leur appliquant un filtre antibruit pour les rendre comparables pixel par pixel.
- **Niveau intermédiaire.** On découpe les images en régions auxquelles l'on attache des propriétés. L'enjeu et la difficulté consistent à trouver des régions qui correspondent à des unités sémantiques de l'image (figure 2), pour que comparer deux images revienne à comparer les propriétés des régions en tenant compte de leur agencement spatial. Par exemple, si l'on recherche un marteau, on voudra trouver une région allongée pour le manche, collée à une autre plus compacte pour la tête. La région « reflet sur le manche », non-porteuse d'information sémantique, ne nous intéresse pas.
- **Haut niveau.** On suppose qu'un système automatique ou opérateur humain a interprété les images en isolant et nommant ses éléments constitutifs. La comparaison se limite à un travail sur des mots-clés.

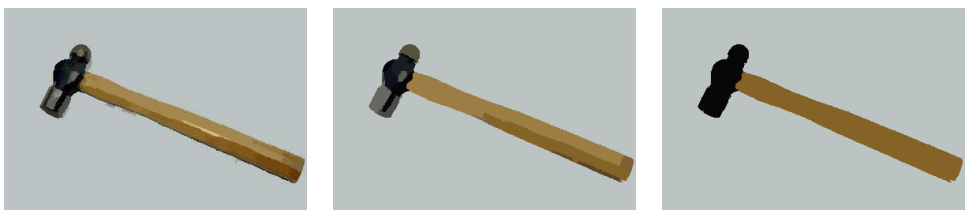

FIG. 2 – À gauche, deux segmentations de la même image. Noter la persistance des reflets et des effets d'éclairage. À droite, la segmentation idéale obtenue manuellement.

Dans cette étude, nous privilégions le niveau intermédiaire, dit également *structurel* car il concerne la structure de l'image, son « squelette spatial ». Cela dans un triple objectif :

- **Rapidité.** La recherche s'effectue sur une quantité à la fois restreinte et discriminante de données.
- **Robustesse.** S'élever au niveau structurel assouplit la comparaison : la notion de sous-objet est robuste aux changements de taille, de couleur ou de position entre image et modèle. Cela de manière prévisible et ajustable : on peut trouver les bons paramètres par le calcul, sans tâtonnement, et facilement les adapter d'une série d'images à l'autre. À l'inverse, les approches basées pixels sont difficiles à paramétrer numériquement et sont parfois très sensibles aux conditions d'éclairage.
- **Automatisation.** On ne demandera pas à l'utilisateur de désigner ni de nommer les sous-objets de son modèle, le moteur devra pouvoir le faire tout seul à partir d'une image du modèle.

1 Travaux connexes

Historiquement, les systèmes de comparaison d'images se sont d'abord contentés de critères globaux. En travaillant sur le signal de l'image toute entière, par le calcul d'un histogramme de couleurs, Swain et Ballard [14] ont obtenu des résultats d'acuité parfois surprenante. Cependant, comme on n'isole pas l'objet du fond, cette approche est très sensible aux conditions d'éclairage et au contenu de l'arrière plan. De plus, elle ne tire aucun parti des informations spatiales sur la forme et la disposition des objets. Cependant ce concept est utilisé par d'autres algorithmes plus évolués, par exemple lors des segmentations.

1.1 Approches basées régions

D'autres systèmes de reconnaissance d'objets s'appuient sur une segmentation préalable de l'image : on regroupe les pixels jugés similaires en une même région. Ensuite, on fusionne les petites régions pour en former de plus grosses. Pour chacune des régions, un certain nombre de descripteurs (forme, couleur, texture) est alors calculé pour servir lors de la comparaison au modèle. C'est l'approche utilisée par Carson et al. [2] dans Blobworld.

Devant l'impossibilité de réaliser une segmentation automatique robuste, c'est-à-dire de toujours parvenir à trouver les régions correspondant exactement aux objets sémantiques, ces approches sont restées limitées. Une amélioration est proposée dans SIMPLIcity par Li et Wang [15], qui consiste à comparer une région d'un modèle à plusieurs régions dans l'image segmentée, afin de réduire les imprécisions dues aux erreurs de segmentation.

Une autre solution consiste à se reposer sur une segmentation manuelle ou assistée comme dans les systèmes QBIC [5] ou IMALBUM [7]. Néanmoins, cette intervention de l'utilisateur pour la segmentation de chaque image est très limitante. Autant nous pouvons autoriser une assistance ponctuelle de l'utilisateur au moment de la constitution du modèle, autant l'analyse des images de la base doit être automatique.

Dans tous les cas, l'erreur de méthode inhérente à la segmentation rend difficile l'exploitation du résultat brut.

1.2 Approches basées formes

D'autres approches considèrent un seul descripteur et l'exploitent pour comparer des objets. Les descripteurs normalisés par MPEG-7, ART [8] pour les régions, CSS [11] pour les contours, sont ainsi d'une efficacité remarquable pour effectuer des recherches dans une banque d'objets détourés. Ils n'exigent que peu d'informations à stocker, et la comparaison de deux objets se ramène au calcul d'une distance entre deux vecteurs.

Cependant, la généricité de ces descripteurs diminue leur précision pour certaines utilisations. Notamment, en reconnaissance d'écriture, les descripteurs de contour et de région font difficilement face aux centaines de manières d'écrire des chiffres. La solution de Belongie [1] consiste à s'appuyer sur le contour des objets, et à calculer la transformation qui déforme le modèle en image. Cette transformation peut être non-linéaire, il s'agit typiquement de minimiser l'énergie nécessaire à la déformation d'un quadrillage. On obtient ainsi une méthode robuste, complémentaire à ART et CSS pour la recherche dans des bases de symboles. Elle permet de trouver des ressemblances structurelles locales entre symboles, même si la forme globale des objets est différente. Le principal avantage de ce procédé est de donner un sens au codage : au lieu de viser la compacité de la représentation, Belongie préfère rendre le codage significatif afin de rendre la comparaison plus robuste.

La limitation fondamentale de ces méthodes est la nécessité d'une présegmentation sémantique : comme ART et CSS ne prétraitent pas les régions qu'on leur soumet, ces descripteurs supposent que la région est déjà détourée et isolée de l'arrière plan. Or il s'agit du problème le plus difficile en reconnaissance d'objets.

1.3 Approches basées pixels

De nombreuses méthodes directement basées pixels sont alors apparues pour pallier les défauts liés à la segmentation. Ainsi, Schneidermann et al. proposent un ensemble de classificateurs bayésiens, fondés sur des descripteurs de type ondelettes [12]. Des résultats de qualité ont été présentés, sur des requêtes de type « visage » ou « voiture ». Dans une approche comparable, Fergus et al. [4] utilisent des masques invariants à l'échelle pour caractériser des zones d'intérêt.

Toutefois, ces approches sont très exigeantes en temps de calcul. De par leur nature statistique, générer un modèle par apprentissage satisfaisant nécessite souvent une collection importante d'images de calibrage. En outre, les paramètres des modèles sont fixés une fois pour toutes dans l'apprentissage, lequel est trop long pour être effectué au moment de la recherche. Cela empêche la création d'un véritable moteur de recherche puisque les requêtes sont fixées d'avance : on est limité à du *clustering* statique.

Une accélération de ces méthodes proposée par Lowe [10] permet d'approcher le temps réel, en considérant les points de fort gradient dans une image en niveaux de gris. Cette approche bas niveau est rapide, au prix d'une perte de souplesse. Elle résiste au changement de point de vue, mais comme les points de fort gradient sont intimement liés aux sources de lumière, un changement des conditions d'éclairage fausse le résultat. Comme les autres approches basées pixels, elle ne prend pas en compte la forme des objets mais discrétise fortement le contour des objets. Cela permet de gérer les occlusions : la perte de quelques points de fort gradient n'est pas rédhibitoire. Par contre le calcul du descripteur ne peut se faire que par un coûteux apprentissage : un opérateur ne peut pas aider la machine en donnant d'entrée de jeu la description du « marteau moyen ».

2 Proposition

Les approches exposées ne parviennent pas, lors de la comparaison des objets identifiés (points ou régions), à détecter et ignorer les artefacts et les éléments sémantiquement négligeables : seul un humain peut faire le tri de manière robuste. D'où la grande sensibilité des méthodes décrites aux effets d'éclairage. Notre démarche consiste à différer le plus possible les décisions effectuées dans ce domaine, afin de conserver une marge de manœuvre confortable tout au long de l'algorithme.

Nous considérons que la segmentation est utile à condition de ne pas garder un seul niveau de segmentation, i.e. une seule partition de l'image en régions. Nous préférons calculer des segmentations à divers niveaux de détail et garder *toutes* les régions rencontrées. Nous construisons une *hiérarchie* de segmentations, de manière incrémentale : en partant d'une sur-segmentation, dont les régions sont petites, nous procédons à un groupement en fusionnant les régions similaires.

Pour effectuer ce regroupement, nous avons choisi d'appliquer le groupement perceptuel [9]. Pour décider de la fusion de deux régions données, nous utilisons les descripteurs perceptuels de Gestalt, déduits d'études sur la vision humaine [6]. Ils reposent sur d'autres critères que ceux de la segmentation (limités à la couleur et la texture), et prennent en compte le fait qu'un objet complexe est souvent composé de sous-objets de couleur différente mais « géométriquement liés » (parallélisme, continuité, etc.).

Conscients de l'efficacité des descripteurs de forme sur des objets clairement délimités, nous les utilisons sur les régions de la hiérarchie de segmentations. En travaillant uniquement sur les régions et non sur les pixels, nous serons à même de comparer des images de manière structurelle, en nous abstrayant de la notion de pixel.

En combinant des descripteurs variés, éventuellement adaptés à une application pratique donnée, nous maximisons la robustesse. Ici, nous utilisons deux jeux de descripteurs : d'une part des descripteurs *de région* considérant la région isolée (sa forme, sa couleur), d'autre part des descripteurs *de structure* prenant en compte la position et la taille de la région par rapport à l'objet entier.

Nous partons donc d'un arbre de régions, dans lequel on cherche à retrouver les éléments d'un modèle, afin de calculer une distance globale entre l'image et le modèle.

Pour maximiser les performances, notre algorithme se découpe en trois phases distinctes :

1. **Découpage** d'une image donnée en régions, à différents niveaux de détail. Cette opération est effectuée une fois pour toutes, le temps réel n'est donc pas indispensable. L'opération consiste en une segmentation basée régions suivie d'un groupement perceptuel.
2. **Comparaison** de la structure de l'image avec celle d'un modèle et calcul d'une distance.
3. **Classement** des résultats, obtenu en calculant la distance du modèle à chacune des images.

L'algorithme de détection des régions d'une image étant soumis aux besoins de l'algorithme de comparaison, nous commencerons par présenter ce dernier avant d'expliquer comment calculer ses données d'entrée.

3 Comparaison d'un modèle et d'une image

3.1 Données d'entrée

- **Un modèle.** Ses données sont les sous-objets $L_1, \ldots, L_n$ qu'il contient, avec leurs propriétés – figure 3(a). On suppose que cette **liste** de sous-objets est discriminante et suffit à l'identification du modèle : deux modèles différents doivent avoir des sous-objets différents. Par contre, dans le cas des objets asymétriques, plusieurs vues donc plusieurs modèles peuvent être nécessaires pour identifier un même objet sous tous les points de vue possibles. Enfin, on suppose qu'un opérateur humain s'est assuré de la qualité des modèles : l'algorithme ne remettra jamais en cause leur découpage en sous-objets.
- **Une image.** L'algorithme ne supposera pas la segmentation effectuée « intelligemment » : il se contente d'un **arbre** de régions $A_1, \ldots, A_m$, donc de sous-objets *potentiels* (figures 3(b) et 4). Si A et B sont les fils d'une région C, cela signifie que l'on a identifié trois sous-objets potentiels : A, B, et leur réunion $C = A \cup B$.

L'objectif est de calculer la distance $|L - A|$.

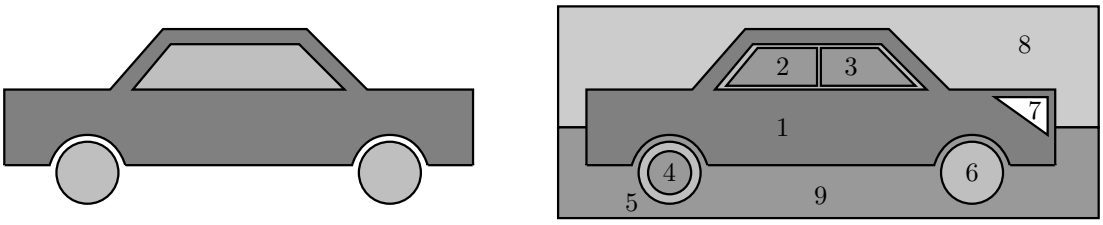

FIG. 3 – (a) À gauche, les sous-objets du modèle. (b) À droite, les régions trouvées dans une image.

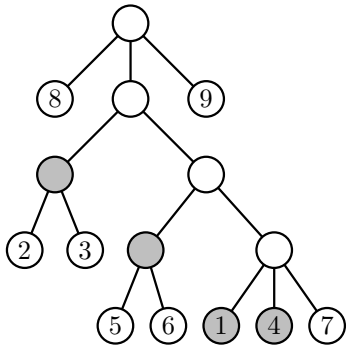

FIG. 4 – L'arbre des régions de l'image. En gris les *nœuds importants*, les régions qui correspondent aux sous-objets du modèle « voiture ».

La figure 4 présente l'arbre de sous-régions optimal. Toute la difficulté consiste à trouver parmi tous ces nœuds ceux qui correspondent à un sous-objet du modèle, que l'on appelle *nœuds importants* (en gris sur la figure). On appelle *nœuds de bruit* ou *nœuds d'arrière plan* les nœuds n'étant et ne contenant pas de nœud important.

Pour résoudre ce problème, l'algorithme fonctionne en trois étapes. Tout d'abord, il isole l'objet de l'arrière plan en trouvant un sous-arbre k contenant tous les nœuds importants et le moins de nœuds de bruit possible. Ensuite, il associe les L_i aux meilleurs A_j possibles en s'aidant des distances entre chaque L_i et chaque A_j pour le sous-arbre k, distances notées $|L_i - A_j|_k$. Enfin, il calcule la distance $|L - A|_k$ entre le modèle et le sous-arbre en s'aidant des distances $|L_i - A_j|_k$ correspondant aux associations trouvées.

3.2 Sous-arbre important

La première étape consiste à délimiter l'objet dans l'image. Dans la figure 3(b), les régions 8 et 9 doivent être ignorées. En pratique, cela revient à considérer un sous-arbre de l'arbre des régions, le plus petit possible, contenant tous les nœuds importants. On l'appellera *sous-arbre important*. Dans la figure 3(b), il s'agit du deuxième fils de la racine, situé entre les nœuds 8 et 9.

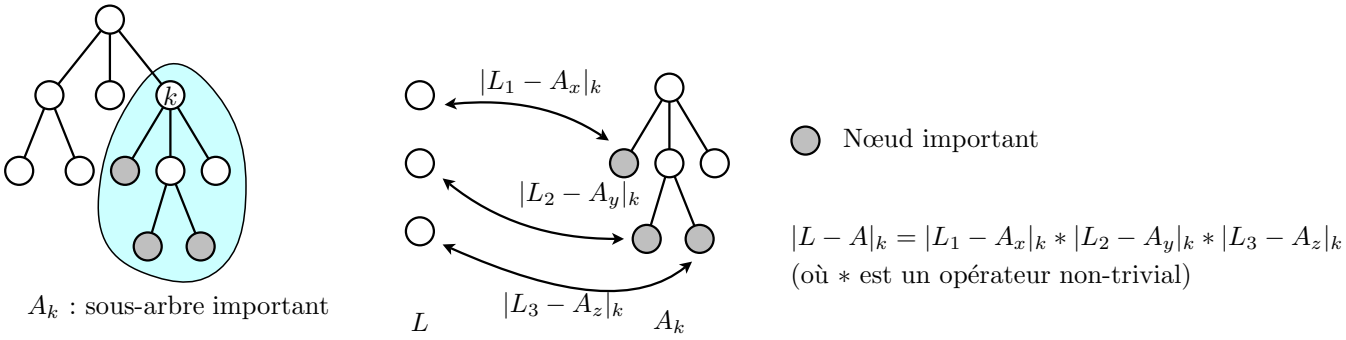

FIG. 5 – Les trois étapes de l'algorithme : détermination du sous-arbre important, choix des meilleures associations, calcul de la distance globale. L est le modèle, A les régions de l'image.

On suppose que deux contraintes sont remplies si l'image correspond au modèle :

- **(Contrainte n° 1)** le sous-arbre important contient *tous* les nœuds importants.
- **(Contrainte n° 2)** le sous-arbre important contient « relativement peu » de nœuds de bruit, donc que les nœuds importants aient un parent commun « assez bas » dans l'arbre.

Comme il est difficile de trouver de manière robuste *le* sous-arbre important, l'algorithme essaye tous les sous-arbres possibles (il en existe autant que de nœuds). Un premier tri est fait à ce niveau : on peut éliminer les sous-arbres dont le ratio de taille (ou l'excentricité) est trop éloigné de celui du modèle : si l'on recherche un objet allongé, le sous-arbre important doit l'être aussi.

3.3 Association des sous-objets du modèle aux régions de l'image

On veut calculer la distance $|L - A|_k$, k étant le sous-arbre important choisi. On suppose savoir calculer la distance $|L_i - A_j|_k$ pour tout (i, j, k). Calculer $|L - A|_k$ revient à trouver un maximum d'associations entre les L_i et les fils A_j de A_k, tout en minimisant les $|L_i - A_j|_k$ qui correspondent. On veut donc un couplage maximal de poids minimal entre les $L_1, \ldots, L_n$ d'une part et les $A_1, \ldots, A_m$ d'autre part, en pondérant les arcs par les $|L_i - A_j|_k$. Les algorithmes connus de couplage dans les graphes bipartis pondérés sont sans secours ici puisque l'arbre casse la structure de graphe. En effet, si l'on a choisi d'associer L_i à A_j, alors il n'est plus possible d'associer les fils et les ancêtres de A_j. Dans l'exemple de la figure 4, il n'est pas possible d'associer à la fois la roue gauche et la jante, car cette dernière serait associée deux fois. Sur la figure 6, les nœuds en gris ne sont plus associables car ils appartiennent à une branche comportant déjà un nœud associé.

À défaut de pouvoir utiliser un algorithme exhaustif qui testerait toutes les associations possibles, une approche gloutonne est acceptable dans notre cas. Elle consiste à effectuer séquentiellement autant d'associations (L_i, A_j) que possible, en choisissant à chaque itération celle qui minimise $|L_i - A_j|$. À la fin, on ajoute une pénalité à la distance $|L - A|$ pour chaque L_i non-associé, car le modèle n'a été reconnu que partiellement. La contrainte n° 1 sur l'arbre des régions assure que si l'image et le modèle correspondent, tous les L_i pourront être associés aux A_j, ce qui exclut toute pénalité si l'approximation gloutonne est valide.

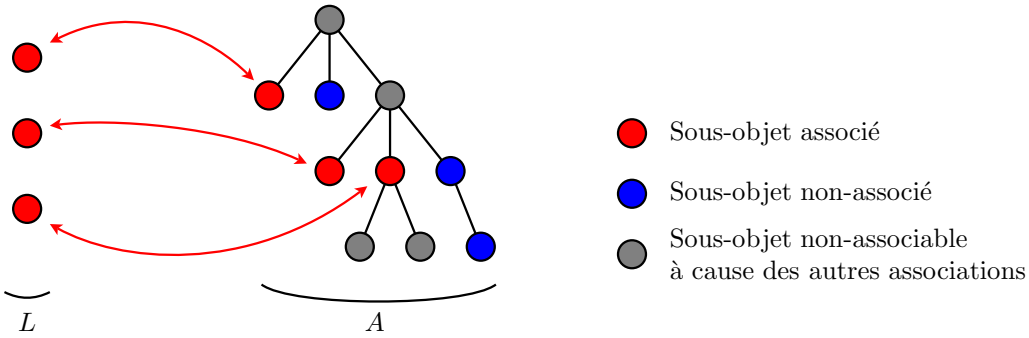

FIG. 6 – Exemple d'association rendue par l'algorithme.

3.4 Descripteurs utilisés

3.4.1 Descripteurs de région, indépendants du sous-arbre important choisi

Pour une région donnée, on stocke les propriétés suivantes :
- **Forme :** description de la région selon ART [8], et du contour selon CSS [11] (cf. annexe 1).
- **Couleur :** ici, on distingue l'image des modèles. Pour une image, il suffit de conserver la couleur actuelle de la région (moyenne et variance). Pour un modèle, il faut indiquer toutes les couleurs possibles, le calcul de la distance prenant la couleur la plus proche entre image et modèle. Il faudrait idéalement calculer un spectre de couleur mais, dans un premier temps, la couleur moyenne et sa variance suffiront.

3.4.2 Descripteurs de structure, dépendants du sous-arbre important choisi

L'objectif est de mesurer la différence entre deux régions en terme de :
- **position :** distance entre les régions, nulle si les régions se recouvrent
- **dimensions :** différence entre la taille relative des régions par rapport à l'objet entier
- **orientation :** critère d'autant plus important que les régions sont allongées

Ces descripteurs dépendent du sous-arbre important choisi, car ils dépendent du placement des régions *par rapport à l'objet*.

On a donc besoin d'une estimation de la forme et de la position de l'objet tout entier, pour mesurer la taille et la position des régions par rapport à lui. Nous utilisons en pratique l'*ellipse englobante* du sous-arbre important (cf. annexe 2), c'est-à-dire l'ellipse ayant les mêmes moments d'inertie que l'objet tout entier, ce qui donne une approximation des dimensions et de la direction de l'objet. Pour comparer deux objets O et O', on calcule leurs ellipses englobantes E et E', et l'on détermine la transformation affine T telle que $E = T(E')$. On compare ensuite les régions $S_1, \ldots, S_n$ de O aux régions transformées $T(S'_1), \ldots, T(S'_{n'})$ (figure 7).

Pour comparer la taille et la position de S_i et S'_j, on calcule l'ellipse englobante de chacun d'eux, et on compare leurs axes (ratio des grands axes entre eux, des petits axes entre eux) et leur centre (distance de l'un à l'autre).

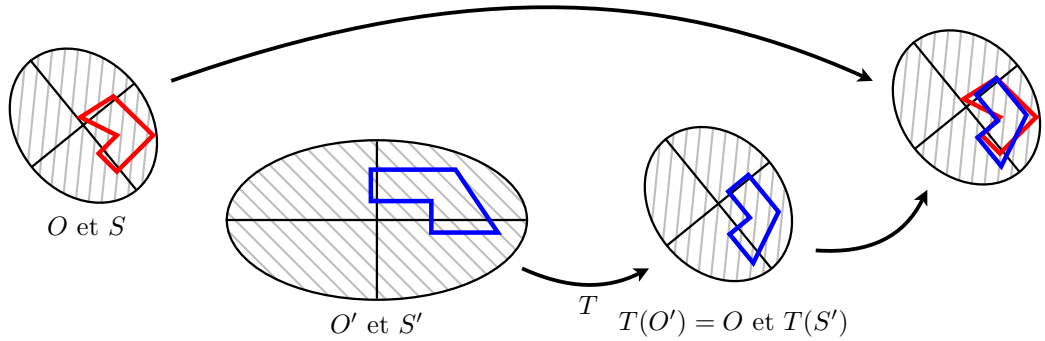

FIG. 7 – Application de la transformation T pour comparer deux objets. Le calcul de $T(S')$ étant mathématiquement difficile, on utilise une approximation (cf. annexe 2).

Cependant, si l'on sait calculer facilement l'ellipse englobante du modèle (c'est l'ellipse englobante de l'union des sous-objets), on ne peut pas faire de même avec l'objet de l'image. En effet, on ne sait pas dire quelles régions correspondent à des sous-objets. Il est difficile de savoir si telle région appartient réellement à l'objet ou bien à l'arrière plan, et donc s'il faut la prendre en compte dans l'ellipse englobante (figure 8).

L'approche retenue pour résoudre ce problème consiste à essayer toutes les ellipses englobantes possibles. Pour éviter de tester les 2^m objets potentiels – c'est-à-dire 2^m ensembles de régions –, on utilise la contrainte n° 2 sur l'arbre des régions : il existe un sous-arbre qui contient tous les nœuds importants et peu de nœuds de bruit. Donc, en considérant tous les sous-arbres du graphe, on est assuré d'en trouver un dont l'ellipse englobante est satisfaisante. Comme il existe autant de sous-arbres que de nœuds, on considérera au plus m ensembles de régions. Pour calculer effectivement l'ellipse englobante, on utilise la région de la racine du sous-arbre, i.e. l'union des régions des feuilles du sous-arbre considéré.

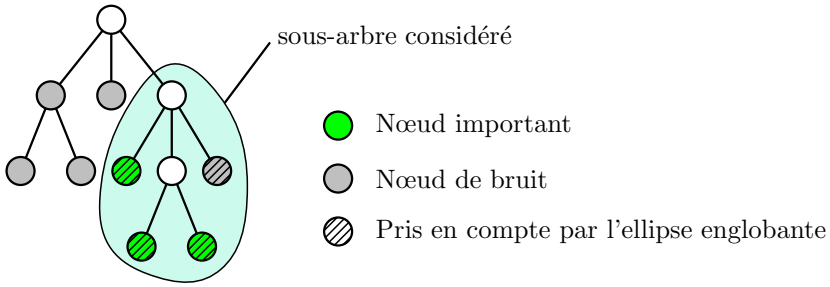

FIG. 8 – Sous-arbre dont la contrainte n° 2 garantit l'existence. Noter qu'une feuille du sous-arbre est un nœud de bruit, et sera prise en compte, à tort, dans l'ellipse englobante.

3.4.3 Combinaison des descripteurs par la méthode de Dempster

Pour calculer la distance entre deux régions, on pourrait se contenter de sommer les distances entre chaque paire de descripteurs. Cependant cette technique n'est pas robuste à l'erreur ponctuelle : si un des descripteurs ne correspond pas, la distance est fortement augmentée. Nous préférons donc utiliser une généralisation de technique probabiliste, les croyances de Shafer [13] (détaillées en annexe 3). On obtient au final une croyance de similarité, que l'on peut assimiler à l'inverse d'une distance sans changer l'algorithme de comparaison.

Le formalisme est utilisé à deux reprises : lors de l'association des sous-objets du modèles aux régions de l'image ($|L_i - A_j|$), et lors du calcul de la distance globale entre le modèle et l'hypothèse d'objet complet ($|L - A|_k$). Dans le premier cas, on fait intervenir des critères correspondant aux descripteurs ART, CSS, position relative et taille relative. À partir de la différence calculée entre les critères de ces deux régions, on détermine une croyance positive ainsi qu'une incertitude pour qualifier la ressemblance des deux régions : la croyance sera forte si les critères se ressemblent alors que l'incertitude sera élevée lorsque les critères sont différents. On combine ensuite ces croyances pour n'en former qu'une seule. Comme il n'y a pas de croyance négative, aucun critère ne rejettera explicitement une association ; par contre la convergence simultanée de *plusieurs* critères est nécessaire pour obtenir une croyance acceptable.

Ensuite, à chaque fois qu'un sous-objet du modèle n'a pas été associé à une région de l'image, nous applique une pénalité. Cela revient à augmenter la croyance dans l'hypothèse de l'inégalité des objets comparés. On procède de même que ci-dessus pour combiner les croyances négatives en une croyance et une incertitude totales.

Pour calculer la croyance globale en la similarité du modèle et de l'image, la méthode de Dempster (annexe 3) permet de combiner la croyance positive totale, la croyance négative totale et les deux incertitudes associées. Un conflit apparaît lorsque les croyances positive et négative totales sont toutes deux élevées, il est alors préférable de ne pas conclure.

4 Segmentation et groupement perceptuel

Le principe est de segmenter l'image en petites régions selon des critères de couleur et de gradient, puis de procéder par *groupement perceptuel* : on fusionne incrémentalement les régions qui se ressemblent selon des critères perceptuels, sans connaissance *a priori* sur les régions à reconnaître. On espère ainsi, pour chaque sous-objet à trouver, passer par une étape où l'on obtient une et une seule région confondue au sous-objet. Point fondamental : il n'est pas nécessaire d'obtenir cette superposition pour tous les sous-objets à la fois, comme avec les algorithmes basés segmentation classiques.

4.1 Segmentation

L'algorithme commence par effectuer une segmentation bas niveau sur le signal, en n'utilisant que la couleur des pixels [3]. Une sur-segmentation est préférable à une sous-segmentation, car la phase suivante de l'algorithme ne pourra que fusionner des régions.

4.2 Groupement perceptuel

L'ensemble des régions ainsi obtenues est ensuite structuré par un graphe planaire complet. La seconde phase de l'algorithme consiste à réduire le graphe en fusionnant les régions perceptuellement proches. L'algorithme de comparaison aura besoin d'un arbre de régions, c'est-à-dire ici de l'historique des fusions effectuées. Pour construire cet arbre, on le construit incrémentalement, à chaque fusion : si les régions A et B sont fusionnées en C, alors l'arbre de régions donnera A et B comme fils de C.

On pondère les arcs du graphe par deux critères :
- **Faisabilité :** évaluer la ressemblance des paires de régions voisines selon plusieurs descripteurs que nous détaillerons. Ce critère décide quelles fusions seront opérées, mais n'influe pas sur l'ordre des fusions.
- **Priorité :** favoriser la fusion des petites régions, afin de conserver des régions de taille similaire. Ce critère, qui dépend de la similarité, ne s'occupe que de l'ordre des fusions et pas de leur faisabilité.

Le critère de priorité est vital : il vise à construire un arbre équilibré. On veut éviter le phénomène d'agrégation : une seule région qui grossirait par accumulation de ses voisines et donnerait un arbre de type peigne difficilement exploitable par l'algorithme de comparaison.

Pour une région donnée, on stocke les propriétés suivantes :
- **Forme :** liste des pixels contenus.
- **Contour :** approximation polygonale du contour.
- **Couleur :** couleur moyenne de la région, et sa variance.

Pour calculer la distance entre deux régions, on utilise les descripteurs de structure suivants (figure 9) :
- **Similarité :** différence de couleur, dans l'espace de couleurs LUV sur lequel, à la différence du RGB classique, l'usage de la distance euclidienne correspond à l'expérience visuelle humaine. Ignorer la composante de luminance L permet d'être moins sensible aux effets d'éclairage (dégradés, ombres).
- **Fermeture :** l'union des deux régions est d'autant plus « fermée » qu'elle ressemble à son ellipse englobante.
- **Continuité :** présence de segments de contour qui se prolongent d'une région sur l'autre.
- **Symétrie :** présence de segments parallèles d'une région sur l'autre.
- **Courbure :** contours lisses (critère de Prägnanz : réduction de l'entropie du contour).

Les descripteurs sont combinés grâce à la méthode de Dempster (annexe 3) pour produire une unique valeur.

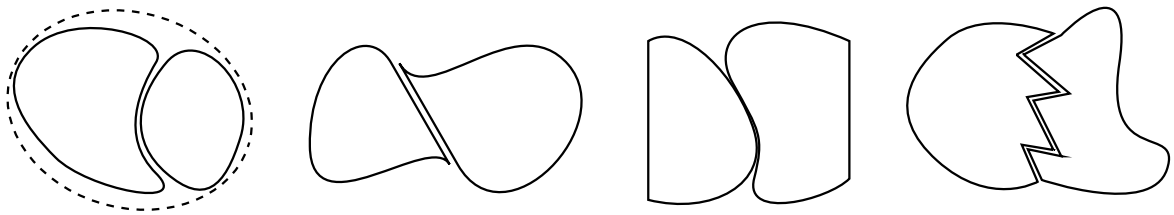

FIG. 9 – Descripteurs structurels de Gestalt.

4.3 Algorithme complet

Segmenter le signal
Créer le graphe planaire complet initial des régions
Pour chaque arc du graphe
    Calculer la faisabilité et la priorité de l'arc
Tant qu'il reste des arcs
    Trouver l'arc de plus forte priorité et de faisabilité non-nulle
        Si non trouvé, arrêter la boucle
    Fusionner en C les deux régions A et B de l'arc
    Ajouter C comme parent de A et B dans l'arbre des régions
    Pour chaque voisin X de A ou de B
        Donner C comme voisin à X et mettre à jour ses faisabilité et priorité

5 Tests

Dans un souci de progressivité, nous testons notre algorithme en deux phases :

1. **Descripteurs de région uniquement.** Les notions structurelles sont mises de côté pour tester les descripteurs de forme (ART, CSS). Les modèles n'auront qu'un seul sous-objet.
2. **Descripteur de région et de structure.** Les modèles auront plusieurs sous-objets afin de mettre à profit la dimension structurelle de notre algorithme. Tous les descripteurs seront mis à contribution, y compris ceux de taille et de position.

5.1 Tests n'utilisant que les descripteurs de région : sans structurel

5.1.1 Test des descripteurs : formes simples

Nous avons tout d'abord étalonné les paramètres non-structurels : ici, les tests ne portent que sur les descripteurs de forme, ART et CSS. Nous avons pris comme modèles des formes géométriques simples, pour les confronter à des images similaires mais bruitées. Une image « bruit » teste la comparaison d'objets très différents.

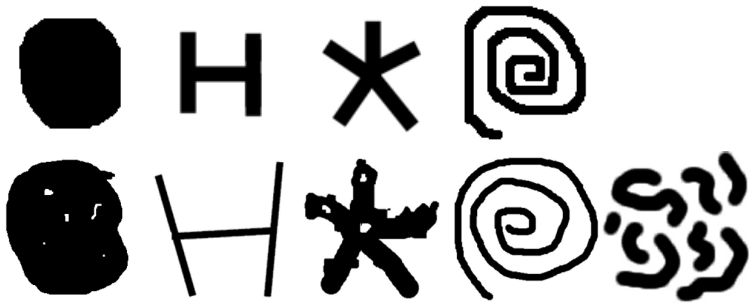

Fig. 10 – En haut, les modèles. En bas, les images.

	Patatoïde	H	Étoile	Spirale
Patatoïde	0,67	—	—	0,62
H	—	0,55	—	—
Étoile	—	0,38	0,60	0,55
Spirale	—	0,48	0,45	0,64
Bruit	0,36	0,47	0,41	0,43

Fig. 11 – Résultat des tests : verticalement les modèles, horizontalement les images. Chaque case contient la similarité calculée lors de la comparaison, comprise entre 0 et 1. L'absence de donnée indique une réponse négative pour cause de trop grande différence mesurée.

Les résultats de la figure 11 sont encourageants. D'une part, les similarités portées par la diagonale sont supérieures à $0,5$ et majorent sensiblement les autres valeurs : elles sont porteuses de sens. D'autre part, l'image « bruit » donne de faibles similarités que l'on peut considérer comme négligeables. Elles résultent principalement du manque de résolution des images : une fois traités et recalés pour ART, les détails deviennent anguleux.

Cette étude simple montre également les limites d'ART et de CSS : l'étoile et la spirale sont considérées comme proches ; la spirale est assimilée à un patatoïde. Les formes approximativement convexes et pleines sont considérées comme similaires : il s'agit d'une erreur de méthode d'ART. Sans le renfort des aspects structurels, les descripteurs de région sont donc insuffisants.

5.1.2 Recherche d'objet dans une bibliothèque

La démarche adoptée est similaire à celle de Blobworld [2] et de SIMPLIcity [15] : le structurel n'intervient pas, puisque le modèle ne comporte qu'un seul sous-objet. Les échantillons montrés sur les figures 12 et 13 affichent un taux de réussite acceptable.

Pour chaque image, l'algorithme calcule un taux de similarité (inscrit sous l'image) et trace l'ellipse englobante de l'objet reconnu. Si le modèle est trouvé plusieurs fois dans l'image, seule l'occurrence la plus ressemblante est retenue. Il serait cependant facile à l'algorithme d'identifier toutes les occurrences du modèle.

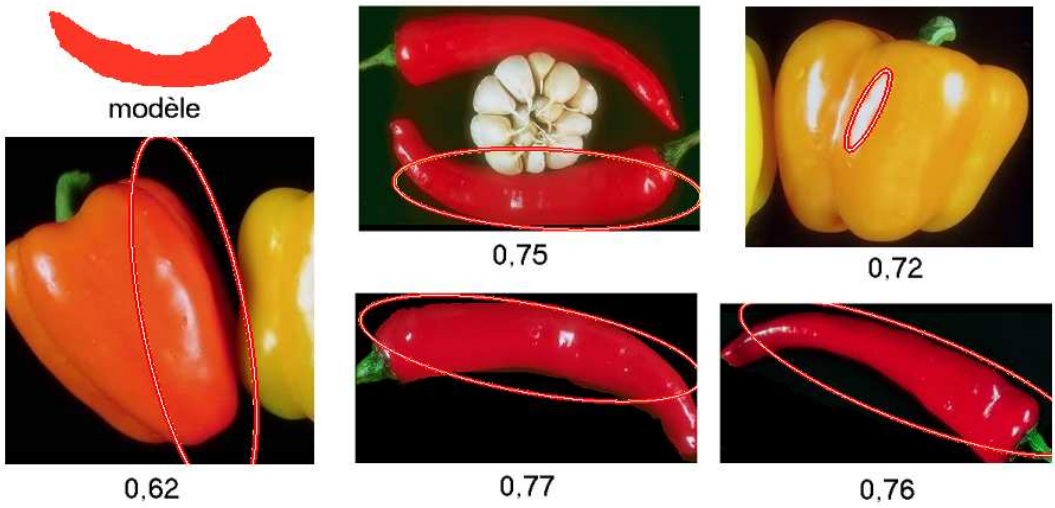

FIG. 12 – Recherche de piments. En haut à droite, l'erreur est due à la non prise en compte de la couleur par notre programme. En bas à gauche, l'erreur est due à la segmentation : le poivron est découpé en deux régions, l'une claire et l'autre foncée ; la seconde a la forme d'un poivron, et est donc reconnue à tort.

Dans la figure 12, l'objet reconnu dans l'image en haut à droite est un artefact. On aurait pu l'éliminer très simplement, à l'aide de critères de taille : le reflet est petit par rapport à l'image. Nous avons décidé de le laisser afin d'illustrer ce défaut de segmentation courant.

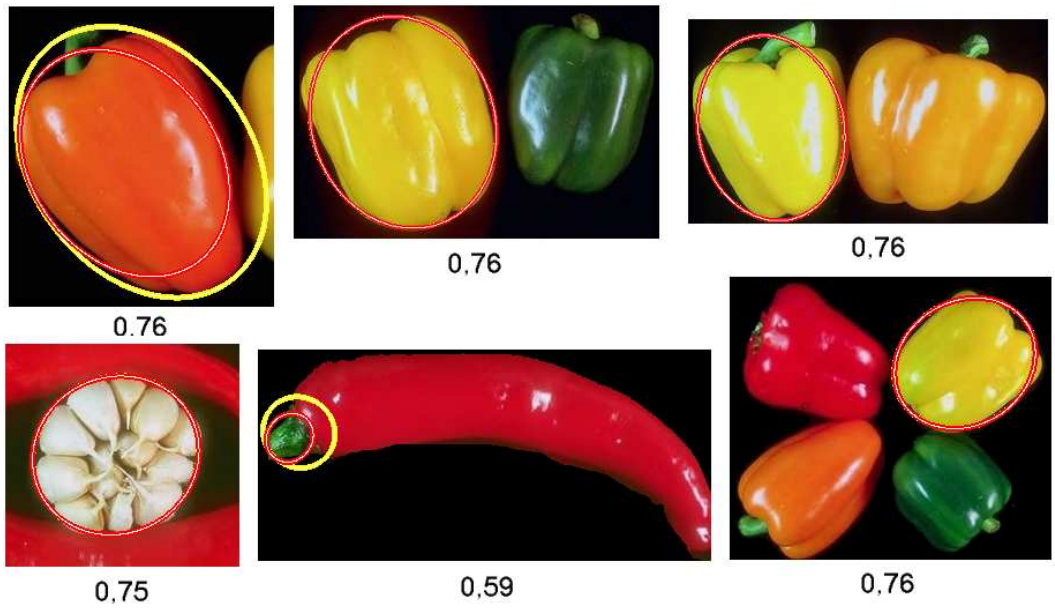

FIG. 13 – Recherche de poivrons. Le modèle, non représenté, est un rectangle aux coins arrondis. Sur les images comportant plusieurs poivrons, seul le meilleur résultat est entouré. En bas à gauche, on note la reconnaissance d'une région complexe mais de la forme recherchée.

5.2 Tests utilisant tous les descripteurs : avec structurel

5.2.1 Reconnaissance de cartes à jouer

Cette fois, nous faisons explicitement appel à l'aspect structurel de notre algorithme. Nous prenons comme modèle un 10 de pique, découpé en plusieurs sous-objets : un pour le fond blanc, et dix autres pour chaque symbole pique. Notez que nous avons volontairement supprimé du modèle les chiffres situés dans les coins, car leur taille et leur complexité les rend difficilement reconnaissables.

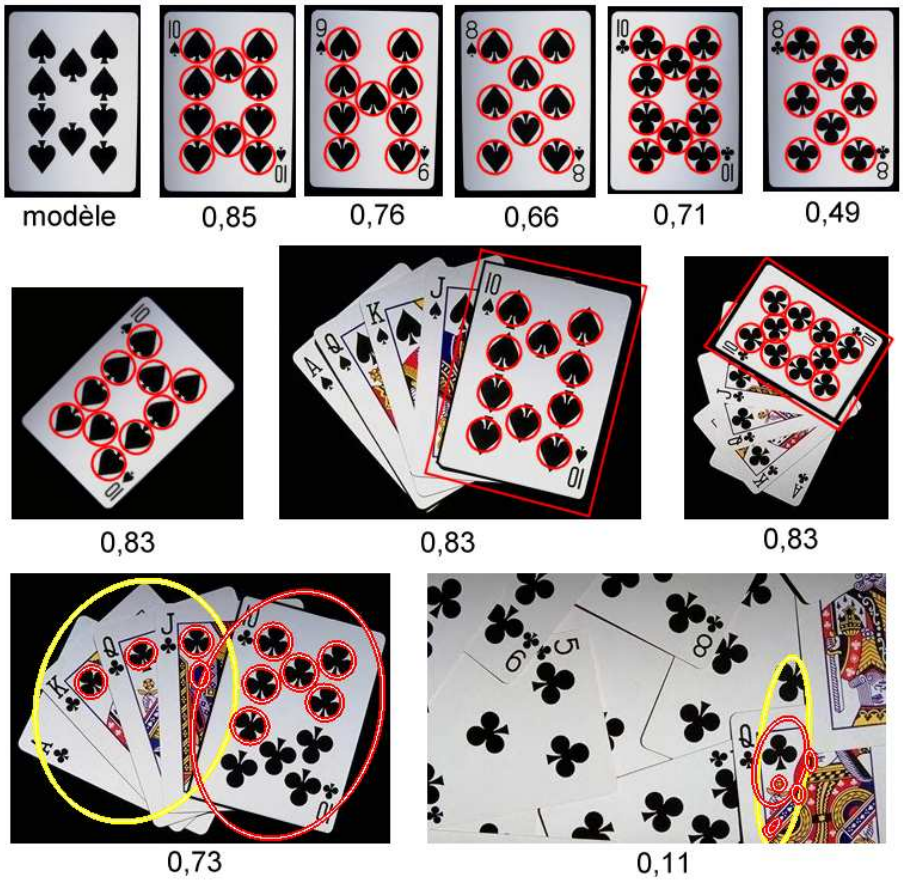

FIG. 14 – Recherche d'un 10 de pique. Le bord des cartes étant difficilement perceptible, un détourage manuel des cartes est nécessaire pour que la segmentation initiale soit correcte.

Les résultats de la figure 14 ont été obtenus pour des réglages relativement souples. Nous avons diminué la pénalité infligée lorsqu'un sous-objet du modèle n'a pas été trouvé dans l'image, sans toutefois l'annuler. Ainsi on constate que le 9 de pique et le 8 de trèfle, pour lesquels il manque respectivement un et deux sous-objets, ont des scores moindres mais significatifs.

Au milieu à droite, l'algorithme a surestimé la similarité. Sans doute la trop faible résolution de l'image d'entrée a-t-elle diminué la précision d'ART : pique et trèfle se ressemblent.

Ce jeu de test illustre très bien l'apport du structurel : l'algorithme est robuste aux rotations et à la présence d'éléments étrangers. Mais surtout, pour une image sans signification (en bas à droite), la similarité calculée est négligeable : l'algorithme ne contente pas de reconnaître localement des éléments du modèle, il exige que la structure du modèle soit respectée.

5.2.2 Recherche d'une structure d'objets dans une bibliothèque

Le principe est similaire au test effectué en 5.1.2 : on recherche un modèle dans une bibliothèque d'objets. La nouveauté est ici la prise en compte de l'aspect structurel : on recherche un modèle composé de deux sous-objets, le manche et l'extrémité d'un outil du type marteau. Les résultats sont présentés dans la figure 15.

Tout d'abord, on constate l'efficacité de l'algorithme, qui a reconnu avec succès les objets similaires à des marteaux, tout en écartant ceux qui n'y ressemblaient pas du tout. Ensuite, on peut voir que l'algorithme accorde du poids aux aspects structurels du modèle, notamment dans le rapport de longueur des deux éléments du modèle : les artefacts sont ignorés, les défauts de segmentation contournés.

Nous avons ensuite recherché les drapeaux à trois bandes verticales parmi les 100 drapeaux d'une catégorie de la base d'images Corel. Les paramètres ont été ajustés de manière à privilégier les descripteurs de structure (taille, position), tandis que les descripteurs de région (ART, CSS) ne compte que pour peu dans le calcul des croyances. L'algorithme de segmentation ayant parfois des difficultés à séparer le ciel des zones claires, nous avons dû colorer en noir le fond de 4 drapeaux.

FIG. 15 – Recherche des objets en forme de marteau parmi des outils.

Les résultats, présentés sur la figure 16, sont très bons. En fixant la barre de croyance à 0,8 nous obtenons aucun faux négatif (les 6 drapeaux tricolores de la base ont été trouvés) pour un seul faux positif.

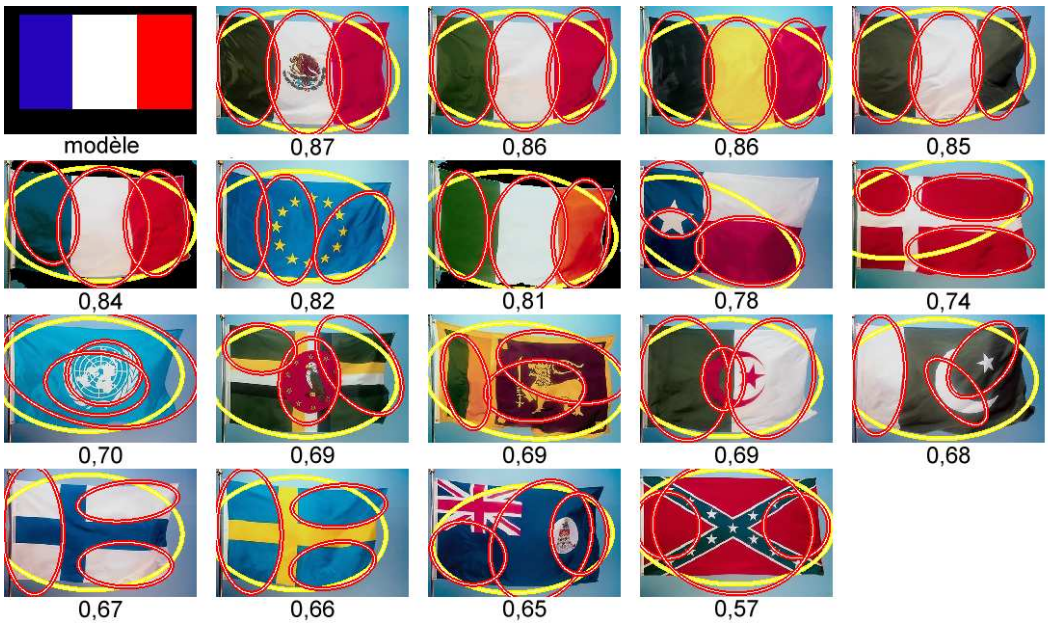

FIG. 16 – Recherche des drapeaux à trois bandes verticales dans une banque de 100 images.

Du côté des performances, l'intégralité du traitement des 100 images de 200×133 pixels (segmentation, calcul des descripteurs, comparaison) prend 30 secondes. Si la segmentation et le calcul des descripteurs des images de la base est effectué en prétraitement et enregistré dans des fichiers, le processus demande environ une seconde.

5.3 Tests sur de gros volumes

L'idée est ici de tester la performance de l'algorithme sur une base de taille plus importante. Nous avons repris les catégories Corel « outils », « drapeaux » et « cartes à jouer » en y ajoutant « objets » et « choses » ne contenant aucun des objets cherchés. Pour chaque modèle, une dizaine d'objets sont pertinents sur les 600 images.

P : nombre total d'images pertinentes parmi les 600 de la base (déterminé humainement)
$P(n)$: nombre d'images pertinentes parmi les n premiers résultats renvoyés par l'algorithme (même remarque)
$r(n) = \frac{P(n)}{P}$: rappel au rang n (nombre de résultats pertinents trouvés)
$p(n) = \frac{P(n)}{n}$: précision au rang n (pertinence des résultats, vaut 1 idéalement)

$r(n)$ étant croissante, il est naturel de tracer la courbe de rappel-précision constituée des points $(r(n), P(n))$. Un mauvais résultat diminue la précision et n'augmente pas le rappel : la courbe d'un bon algorithme aura donc la forme d'une marche d'escalier. Noter que le rappel augmente au détriment de la précision lorsque l'on assouplit les critères de comparaison, et inversement. La courbe de rappel-précision constitue donc une bonne évaluation des algorithmes, en considérant à la fois leur nombre de faux négatifs (rappel) et de faux positifs (précision).

La figure 17 présente les courbes de rappel-précision obtenues. L'image « marteau » finit par trouver environ 75 % des résultats attendus, pour un seul faux positif. Ce dernier étant en seconde position, il grève fortement la courbe de rappel précision. L'image « 10 de pique » affiche de bons résultats : pour une précision toujours supérieure à 75 %, elle trouve progressivement 80 % des résultats attendus.

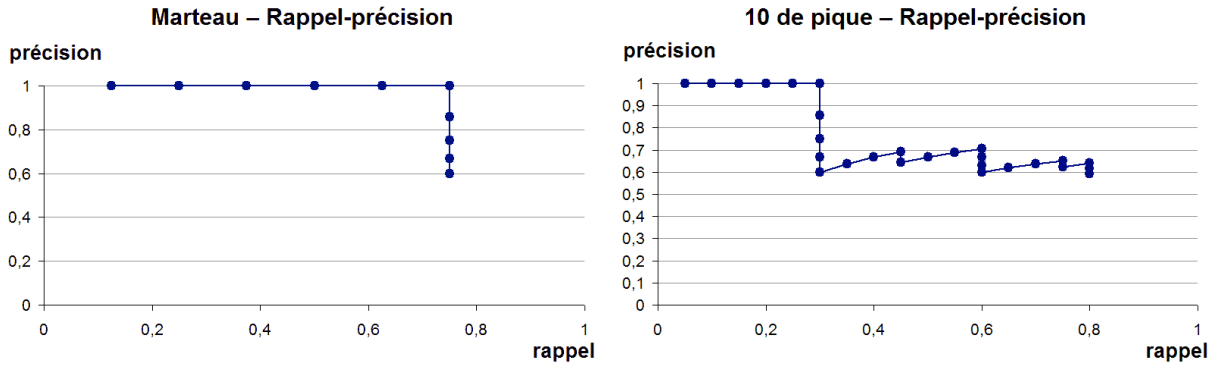

FIG. 17 – Courbes de rappel-précision sur une base de 600 images.

5.4 De l'importance des paramètres

Les courbes présentées ci-dessus n'ont pu être obtenues qu'après un paramétrage assez minutieux du programme. La légère diminution du nombre de faux positifs entre premiers et derniers tests est à leur crédit. Ces paramètres concernent principalement deux critères :

- L'importance accordée au structurel par rapport au non-structurel. Le structurel est par exemple très important pour les drapeaux (taille et position des bandes), tandis que la forme n'est pas pertinente à cause des déformations induites par le vent. Par contre, pour la recherche d'une carte à jouer, il faut essayer de faire la différence entre un pique et un cœur, même s'ils ont la même taille et sont situés au même endroit.
- L'importance et la souplesse de chaque descripteur. Ces critères sont fondamentaux pour la recherche des marteaux : pertinence de la différence de taille, variations d'angle autorisées, etc.

Tous ces critères font partie intégrante du modèle car la valeur brute des descripteurs est insuffisante : il faut une indication sur leur variance et leur pertinence. Noter que les paramètres pourraient être obtenus par des procédés statistiques, mais que leur nombre relativement restreint et leur caractère intuitif les rend accessibles à un opérateur humain, contrairement aux méthodes statistiques évoquées en introduction.

6 Moteur de recherche

Jusqu'à présent, nous n'avons considéré qu'une unique comparaison d'objets. Cette méthode est aisément adaptable à un petit moteur de recherche, de quelques dizaines ou centaines d'images : il suffit de lancer l'algorithme de comparaison sur chacun des modèles de la base de données. Si l'on souhaite rendre notre algorithme accessible aux bases de données de taille importante, il devient nécessaire d'optimiser certains aspects.

6.1 Accélération de la recherche du nœud racine servant au calcul de l'ellipse englobante de l'image

Dans l'algorithme de comparaison, nous testons m ellipses englobantes possibles dans l'arbre de régions (une ellipse par nœud). Nous n'avons pas précisé l'ordre de parcours des nœuds de l'arbre, car en théorie il faudrait tous les examiner. En pratique, il est intéressant de trouver rapidement le k qui minimise $|L - A|_k$ car cela permet d'arrêter immédiatement la recherche.

Pour optimiser notre algorithme, nous allons tirer une nouvelle fois partie de la contrainte n° 2. affirmant que si l'image correspond au modèle, alors il existe un sous-arbre A_{k_0}, proche du modèle, contenant tous les nœuds importants. L'idée est de chercher k_0 par récurrence, en partant du sommet de l'arbre.

Une marche en profondeur, consistant à diriger la recherche dans le fils ayant la plus petite distance, permet d'obtenir une complexité de $O(\log m)$. Cela permet de gagner un facteur 10 pour un arbre d'une soixantaine de nœuds, pour un surcoût de méthode pratiquement nul : le gain n'est pas du tout négligeable. Cependant le risque d'erreur est très important.

Un parcours en largeur, malgré une complexité linéaire dans le pire cas, s'avère efficace en moyenne. L'idée est donc de parcourir récursivement tous les fils en commençant par ceux ayant les plus petites distances. On peut élaguer l'arbre à l'aide d'heuristiques et obtenir ainsi de meilleurs performances que l'algorithme exhaustif.

6.2 Recherche en temps indépendant de la taille de la base de données grâce à une table de hachage

Jusqu'à présent, l'algorithme parcours un par un les images de la base de données, d'où une complexité linéaire en fonction de la taille de la base. Pour réduire cette complexité, on peut utiliser une table de hachage et réduire l'ensemble des images à comparer à uniquement celles susceptibles d'entraîner une réponse positive.

Rechercher les images correspondant à un modèle revient – en partie – à trouver les images qui, pour un sous-arbre important bien choisi, ont des régions de même taille, position et angle que chacun des sous-objets $L_1, \ldots, L_n$ du modèle L. Pour chaque image, et pour chaque sous-arbre important potentiel, on indexe toutes les régions du sous-arbre, d'après leur taille, position, et angle par rapport à l'ellipse englobante du sous-arbre, et d'après le ratio de taille du sous-arbre.

Pour trouver les images correspondant au modèle, on commence par récupérer pour chaque L_i l'ensemble des sous-arbres de ratio de taille identique à celui de L, et contenant une région de même placement que L_i. On calcule leur intersection : cela donne l'ensemble des sous-arbres similaires au modèle *au niveau structurel seulement*. Ensuite, on applique le procédé classique : parcours linéaire de tous les sous-arbres, et comparaison avec l'algorithme complet pour prendre aussi en compte la forme des sous-objets.

La clé de hachage prend en compte deux critères : le ratio de taille du sous-arbre important, la taille et la position de la région par rapport au sous-arbre important. On est obligé de quantifier ces valeurs pour ne pas perdre la notion de proximité (deux tailles presque – mais pas – identiques par exemple). L'espace de chaque valeur est donc partitionné en un certain nombre de sous-ensembles, et l'on indexe non pas la valeur elle-même mais le numéro du sous-ensemble auquel elle appartient.

Conclusion et perspectives

Le structurel et les critères perceptuels, déterminants dans l'algorithme exposé, se sont avérés très bénéfiques :

- **Rapidité.** Les prétraitements appliqués aux images de la base sont relativement peu coûteux ; l'algorithme de recherche est rapide car travaille sur la structure abstraite des images, indépendante de leur nombre de pixels, et souvent limitée à une centaine de régions. La possibilité d'utiliser une table de hachage rend l'algorithme accessible aux très grandes bases d'images, au prix toutefois d'une perte de souplesse.
- **Simplicité.** L'abstraction du structurel permet de spécifier l'objet à rechercher en piochant dans une base de modèles prédéfinie, mais aussi en le dessinant à l'aide de primitives graphiques simples. On peut imaginer un moteur de recherche permettant d'utiliser un logiciel de dessin vectoriel minimaliste pour entrer les requêtes.
- **Paramétrabilité.** Les modèles ne sont pas uniquement décrits par des formes géométriques, ils ont besoin de paramètres numériques simples, assimilables à une sorte d'écart-type des formes géométriques. Malgré leur nombre, ces paramètres sont intuitifs et peuvent être intégrés au logiciel de dessin vectoriel de manière naturelle lorsque la valeur par défaut n'est pas assez discriminante.

- **Souplesse.** L'algorithme est générique et rapidement adaptable à des applications spécifiques données. Sa modularité (découpage en phases distinctes) permet d'optimiser certaines étapes sans remettre en question l'ensemble. Cela est visible dans le code même du programme réalisé dans le cadre de ce stage.
- **Réduction du fossé sémantique.** L'algorithme ne se contente pas de calculer un taux de similarité entre un modèle et les images d'une base, il localise précisément les sous-objets du modèle dans chaque résultat. Par exemple, dans chaque image assimilée à un marteau, on saura où se trouve le manche et la tête.

Cependant, pour que l'algorithme soit efficace, les images doivent avoir une composante structurelle forte. D'une part, elles doivent comporter des régions clairement délimitées. Il s'agit de la contrainte la plus restrictive : en pratique, notre algorithme ne peut traiter efficacement que des objets simples, en aucun cas des scènes complexes dans leur ensemble. Tout au plus sera-t-il possible d'identifier un objet simple noyé dans une scène complexe. D'autre part, ces régions doivent correspondre à des éléments sémantiques discriminants et communs à tous les objets de la classe. Il est ainsi difficile de reconnaître en toute généricité les êtres humains d'un cliché, car le grand nombre de postures possibles ne permet pas de concevoir un modèle unique.

Actuellement, seul un opérateur humain peut créer les modèles. Même si le procédé reste simple, on peut vouloir générer automatiquement des modèles pour de nombreuses classes préexistantes, disposant chacune d'un grand nombre d'images. On pourrait utiliser une méthode statistique, comme le font les algorithmes bayésiens mentionnés en introduction. Cependant un modèle structurel comporte de nombreux paramètres et un nombre initialement inconnu de sous-objets, rien ne dit que les algorithmes fondés sur les points caractéristiques seront facilement transposables.

Dans le même ordre d'idées, on pourrait améliorer l'algorithme en vue d'effectuer du *clustering* : on dispose d'une base d'objets variés qu'il faut répartir en catégories, lesquelles ne sont pas connues au départ. L'algorithme présenté est inutilisable car exige qu'on lui fournisse d'entrée un modèle par catégorie. On peut imaginer une méthode fondée à la fois sur des critères perceptuels haut niveau et sur les régions des images.

L'objectif était d'utiliser les techniques de segmentation et de groupement perceptuel pour exploiter les aspects structurels des images à comparer. Il est atteint : j'ai conçu un algorithme de recherche sur des images présentant des caractères structurels. De plus, grâce à sa relative simplicité, j'ai pu le programmer effectivement, de manière efficace, dans un délai raisonnable et sans rencontrer de difficulté majeure. Aucune optimisation particulière n'est nécessaire pour obtenir de bonnes performances, en partie grâce à la réutilisation d'algorithmes connus et intrinsèquement efficaces, certains disposant déjà d'implémentations rapides.

La réalisation du programme a donné lieu à la programmation en C++ d'un *framework* de manipulation d'images adapté à la reconnaissance d'objets. J'ai intégré (segmentation) ou réécrit (calcul des descripteurs ART et CSS, groupement perceptuel, ellipses englobantes, etc.) les implémentations de plusieurs algorithmes dans une architecture modulaire, afin de faciliter la future évolution du projet. Un système de fichiers de configuration permet de paramétrer le programme au *runtime*, sans recompilation.

Ce programme a permis d'effectuer des tests sur 600 images et de produire des courbes de rappel-précision, une procédure courante pour estimer l'efficacité d'un algorithme de ce type. J'ai donc pu confronter mon algorithme à ceux déjà publiés en me fondant sur des critères reconnus comme significatifs.

Ce stage m'a donc permis de toucher à toutes les étapes d'un projet : lecture et analyse de la bibliographie, focalisation sur un sous-problème, spécification d'une nouvelle méthode de résolution, conception de l'algorithme correspondant, programmation effective des algorithmes (principal et annexes) et d'une interface par fichiers textuels, test sur des données réelles de taille non-anecdotique, analyse des résultats, rédaction d'un rapport exhaustif interne au LIRIS, et enfin, dans les semaines à venir, rédaction d'une publication.

Remerciements

Je tiens à remercier pour l'aide qu'ils m'ont apportée tout au long du stage :
- Nicolas Zlatoff, pour son encadrement et son suivi régulier, sa « présentation par l'exemple » du milieu de la recherche, et les nombreuses aides qu'il a pu m'apporter ;
- Bruno Tellez, pour m'avoir permis d'effectuer ce stage et de le poursuivre un mois supplémentaire, et pour m'avoir accordé une grande liberté quant au choix de mon sujet ;
- David Cœurjolly, pour les coups de pouce en maths ;
- Clément, dont les livres et les codes source ont grandement facilité la réalisation du programme ;
- Brice, Céline, Loris, et tous les stagiaires ou thésards, pour la bonne ambiance malgré l'absence de clim.

Annexe 1 : descripteurs ART et CSS

Angular Radial Transform : ART

ART [8] décrit une région partir de la distribution 2D de ses pixels, de manière invariante à la taille, la rotation et la translation. C'est une transformation complexe unitaire définie sur un disque unité, qui consiste en une projection de l'image sur des fonctions de base, dans un repère polaire. Ainsi, on définit $F_{m,n}$, les coefficients ART d'ordre m et n d'une image f, par :

$$F_{m,n} = \int_0^{2\pi}\int_0^1 \frac{1}{2\pi} \mathrm{e}^{jm\theta} R_n(\rho) f(\rho,\theta)\, \rho \,\mathrm{d}\rho \,\mathrm{d}\theta$$

La composante radiale étant : $R_n(\rho) = \begin{cases} 1 & n = 0 \\ 2\cos(\pi n \rho) & n \neq 0 \end{cases}$

La série des modules des coefficients $F_{m,n}$, normalisés, constitue le descripteur ART. L'invariance en rotation est obtenue grâce à l'utilisation du module. L'invariance à l'échelle est obtenue en normalisant les modules des coefficients par celui d'ordre 0 ($m = n = 0$). La norme MPEG-7 préconise d'utiliser 12 fonctions angulaires et 3 fonctions radiales, soit $m = 12$ et $n = 3$. Nous conserverons ces valeurs pour notre travail. La figure 18 représente les parties réelles des fonctions de bases étudiées.

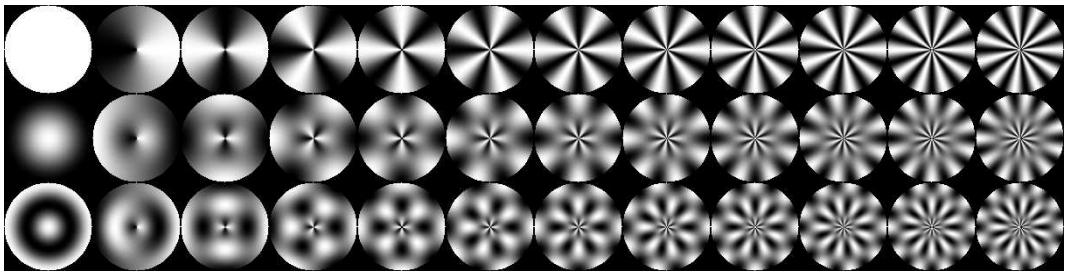

FIG. 18 – Partie réelle des fonctions de base ART étudiées. En colonnes les m, en lignes les n.

Enfin, toujours comme préconisé dans MPEG-7, la distance entre deux objets est calculée par la norme 1.

Curvature Scale Space : CSS

CSS [11] décrit un contour fermé, de manière invariante à la taille, la rotation et la translation. Plus précisément, il considère les positions des points d'inflexion du contour lors d'une série de filtrages gaussiens. À mesure que la largeur du filtre augmente, les inflexions non significatives sont éliminées et le contour devient plus lisse. Au contraire, on considère que les points d'inflexion restants sont représentatifs de la forme (figure 19(a)).

Les points d'inflexion du contour sont ensuite déduits des zéros de la courbure. Le résultat de ce traitement par filtrages successifs est la création d'une image d'empreinte (figure 19(b)), qui présente la liste des points d'inflexion au cours des filtrages, en fonction de leur abscisse curviligne le long du contour.

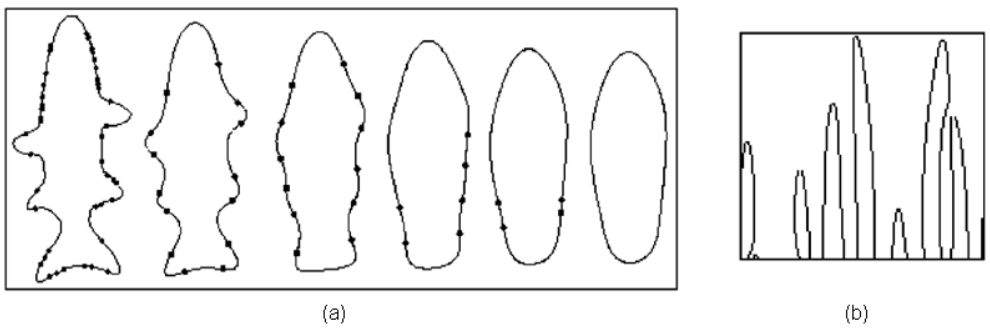

FIG. 19 – Évolution d'un contour au gré des filtrages gaussiens (a) et représentation CSS associée (b).

La mesure de similarité entre un contour issu d'une image R et un contour issu d'une image tierce I consiste en une mesure L_2 entre les pics appariés, modifiée par une pénalité pour chaque pic non associé.

$$D_{CSS}(R,I) = \sum_1 \Big((x_{R,i} - x_{I,j})^2 + (y_{R,i} - y_{I,j})^2 \Big) + \sum_2 y_i^2$$

Avec $\sum_1$ désignant la somme sur tous les pics associés et $\sum_2$ la somme sur tous les pics non-associés.

Annexe 2 : manipulation des ellipses englobantes

Obtention d'une ellipse englobante

On se donne un ensemble de points, dont on connaît les moments d'inertie. On en déduit par des formules simples les moments d'ordre 1 et 2, puis les moments d'inertie d'ordre 2 et enfin les paramètres de l'ellipse.

Pour toute ellipse E, on note $E = E(E_O, E_a, E_b, E_\theta)$ où les éléments du n-uplet sont respectivement le centre, le grand axe, le petit axe et l'angle de l'ellipse.

Transformation d'ellipses

On se donne deux ellipses, E et F, où E est l'ellipse englobante d'une région contenant une sous-région dont l'ellipse englobante est F. On considère donc que F est une sous-ellipse de E. On se donne une ellipse de référence E', et on calcule la transformation f telle que $f(E) = E'$. On désire calculer $F' = f(F)$.

Le calcul de f est simple : c'est la composition de la translation de vecteur $\overrightarrow{E_O E'_O}$, de la rotation d'angle $E'_\theta - E_\theta$, et de l'affinité de rapports $\dfrac{E'_a}{E_a}$ et $\dfrac{E'_b}{E_b}$ selon les axes de E' :

$$f = A\left(\frac{E'_b}{E_b}, E'_\theta + \frac{\pi}{2}\right) \circ A\left(\frac{E'_a}{E_a}, E'_\theta\right) \circ R(E'_\theta - E_\theta) \circ T\left(\overrightarrow{E_O E'_O}\right)$$

L'application de T et de R à une ellipse est simple également. Pour $E = E(P, a, b, \theta)$, on a :

$$T_{\vec{v}}(E) = E(P + \vec{v}, a, b, \theta)$$
$$R_{\Delta\theta}(E) = E(P, a, b, \theta + \Delta\theta)$$

Par contre, l'image d'une ellipse d'angle quelconque par une affinité d'axe quelconque est très difficile à calculer. Il est déjà ardu de prouver que l'image d'une ellipse par une affinité quelconque est bien une ellipse, car les angles et les axes changent selon des proportions non-linéaires. La détermination des paramètres de l'ellipse image est, on le devine, encore plus difficile. Ce problème étant actuellement ouvert du moins sans solution simple connue, nous nous contenterons d'une approximation. Nous supposerons que E et E' sont de ratio similaire, c'est-à-dire que $\dfrac{E_a}{E_b} \approx \dfrac{E'_a}{E'_b}$, donc $\dfrac{E_a}{E'_a} \approx \dfrac{E_b}{E'_b}$. En pratique, dans nos applications, c'est le cas.

On peut alors fusionner les deux affinités en une homothétie de centre E'_O et de rapport $\lambda = \sqrt{\dfrac{E_a\, E_b}{E'_a\, E'_b}}$ pour que l'aire de $E'_{\text{approchée}}$ soit égale à celle de E'_{exacte}, sachant que les ellipses se comportent bien avec les homothéties :

$$H_{P,\lambda}(E) = E\big(H(E_O), \lambda a, \lambda b, \theta\big)$$

On a donc au final :

$$f = H\left(E'_O, \sqrt{\frac{E_a\, E_b}{E'_a\, E'_b}}\right) \circ R\left(E'_\theta - E_\theta\right) \circ T\left(\overrightarrow{E_O E'_O}\right)$$

Distance entre ellipses

Pour améliorer le groupement perceptuel, nous avons envisagé de comptabiliser la distance spatiale entre les deux régions dont on considère la fusion. Si l'on reste au niveau pixel, cette distance est nulle, car les régions considérées sont toujours voisines. Cependant, d'une part nous pouvons considérer un graphe d'adjacence de degré strictement supérieur à 1, d'autre part la distance pixel peut être très inexacte à cause des artefacts. D'où l'idée d'une distance entre les ellipses englobantes : on veut déterminer la plus petite distance séparant un point de l'ellipse de la première région d'un point de l'ellipse de la seconde. La distance est nulle si les ellipses s'intersectent.

Malheureusement, le sous-problème consistant à calculer la distance d'un point à une ellipse est très difficile. Les méthodes trouvées sont soit impossibles à résoudre dans le cas général (équations polynômiales d'ordre 4), soit numériques (convergence d'une suite). Notre problème, étant plus général, ne peut donc qu'être encore plus difficile. Cependant nous n'avons besoin que d'une approximation de la distance entre deux ellipses : nous pouvons nous contenter de discrétiser chaque ellipse en une dizaine de points, et de calculer la plus petite distance entre ces deux ensembles de points.

Observation : le problème de la transformation d'ellipses est au moins aussi difficile que celui de la distance entre un point et une ellipse. En effet, si on savait résoudre la transformation d'ellipses, il suffirait d'appliquer au point et à l'ellipse une transformation qui rendrait circulaire l'ellipse, puis de calculer la distance entre le point et le centre du cercle, à laquelle on retrancherait le rayon du cercle. Conclusion : nous obtenons confirmation de la difficulté du problème de la transformation d'ellipses, sachant que l'étendue de la littérature relative au calcul de la distance entre ellipses en accrédite la difficulté.

Annexe 3 : théorie de Dempster-Shafer

La théorie de Dempster-Shafer est une théorie probabiliste de la preuve. Elle est spécialement conçue pour combiner des points de vue différents sur un jeu d'hypothèses, ce qui est particulièrement adapté à notre cas. Ici, les points de vue proviennent des différents descripteurs utilisés pour comparer les régions ou les structures. Le formalisme nous permet d'en déduire un point de vue unique, qui prend en compte tous les autres.

Soit Θ un ensemble d'hypothèses $\{H_1, H_2, \ldots, H_n\}$ mutuellement exclusives, appelées cadre de discernement. L'ensemble des parties de Θ est noté 2^Θ. Une fonction $m : 2^\Theta \longrightarrow [0\,;1]$ est appelée jeu de masses lorsque :

$$m(\varnothing) = 0$$
$$\sum_{A \subset \Theta} m(A) = 1$$

Pour chaque partie A de Θ, $m(A)$ est appelée la masse de probabilité portée en A. Elle représente la croyance que quelqu'un porte *exactement* en A. En général, cela ne correspond pas à la croyance *totale* que l'on porte en A. En effet, celle-ci est obtenue en sommant les croyances exactes (ou masses de probabilité) sur tous les événements B qui impliquent A. On définit ainsi $Bel(A)$, croyance totale en A, par :

$$Bel(A) = \sum_{B \subset A} m(B)$$

La fonction $Bel : 2^\Theta \to [0\,;1]$ ainsi construite est appelée fonction de croyance. En pratique, on peut manipuler indifféremment des jeux de masses ou des fonctions de croyance.

La règle de combinaison de Dempster permet de combiner plusieurs fonctions de croyance Bel_i sur un même cadre de discernement, pour en déduire une nouvelle fonction de croyance prenant en compte chaque Bel_i.

Le formalisme est utilisé à deux niveaux : lors de l'association des sous-objets du modèle aux régions de la requête ($|L_i - A_j|$) d'une part, et lors du calcul de la distance globale entre le modèle et l'hypothèse d'objet complet ($|L - A|_j$) d'autre part. Concernant le premier cas, à savoir l'association d'un sous-objet du modèle à une région de la requête, on distingue trois hypothèses :

- $\underline{O_{ij}}$: le sous-objet j de la requête est associé à la région i du modèle ;
- $\overline{O_{ij}}$: le sous-objet j de la requête n'est pas associé à la région i du modèle ;
- incertitude résiduelle Θ.

Cinq jeux de masses seront utilisés, pour représenter respectivement les descripteurs ART, CSS, position relative, taille relative et différence d'angle. Chacun de ces jeux de masses pourra engager une croyance sur deux hypothèses seulement : O_{ij} et Θ (incertitude résiduelle).

La combinaison de deux jeux de masses est illustrée en figure 20(a). Elle donne lieu à la création de deux hypothèses : O_{ij} ou Θ. Ainsi, aucun critère ne peut être interprété de manière à rejeter explicitement une association. En revanche, la convergence simultanée des critères est exigée pour disposer d'une croyance maximale en l'association. La croyance résultante en O_{ij} est donnée par :

$$m(O_{ij}) = m_1(O_{ij})m_2(O_{ij}) + m_1(O_{ij})(1 - m_2(O_{ij})) + m_2(O_{ij})(1 - m_1(O_{ij}))$$

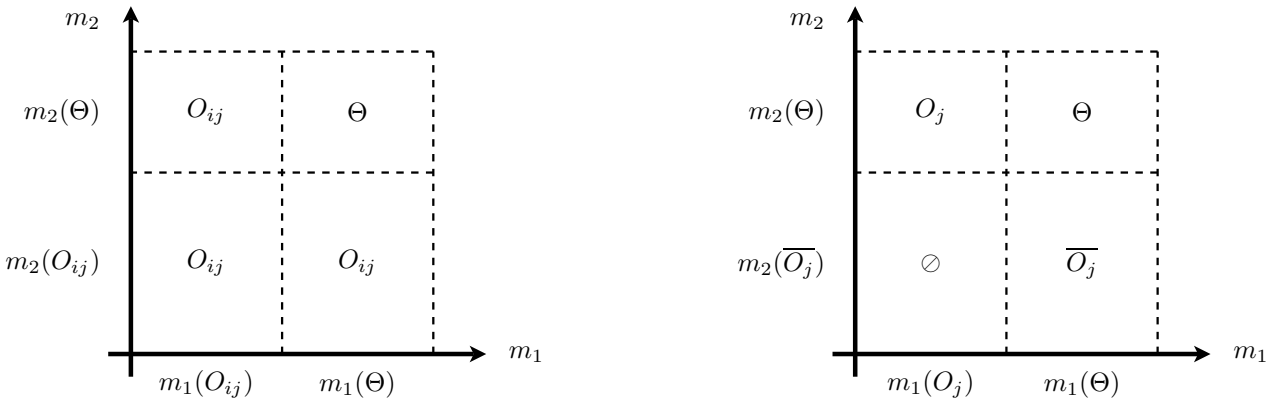

FIG. 20 – (a) Exemple d'application de la règle de Dempster pour la distance globale région/objet complet. (b) Exemple d'application de la règle de Dempster pour l'association sous-objet/région. L'aire des zones est proportionnelle à la croyance du critère associé.

Notons que, du fait du cadre de discernement choisi, il n'y a pas de conflit entre nos hypothèses ($k = 0$). Les résultats de la combinaison sont donc toujours représentatifs.

Examinons maintenant le deuxième cas, à savoir le calcul d'une distance entre une hypothèse d'objet complet dans la requête et l'objet d'un modèle. O_j et $\overline{O_j}$ ont été calculés, il faut maintenant évaluer numériquement le taux d'égalité des deux objets comparés. On considère donc trois hypothèses :
- O_j : la région j de la requête est associée à l'objet complet du modèle ;
- $\overline{O_j}$: la région j de la requête n'est pas associée à l'objet complet du modèle ;
- incertitude résiduelle Θ.

La situation peut être illustrée graphiquement par la figure 20(b). Cette fois, les jeux de masse sont différents :
- m_1 est alimenté par les associations (L_i, A_j), qui répartissent leur croyance sur deux hypothèses : O_j ou Θ, pour confirmer l'hypothèse O_j.
- m_2 est modifié pour chaque L_i qui n'a pas pu être associé à un A_i : cela crée un jeu de masse qui répartit sa croyance sur deux hypothèses : $\overline{O_j}$ ou Θ, pour contredire l'hypothèse O_j.

Notons que cette fois, un conflit entre les hypothèses est susceptible d'apparaître. Dans le cas d'un conflit trop important, il sera judicieux de ne pas conclure quant à la présence de l'objet.

Références

[1] S. Belongie, J. Malik, J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24(4), pp. 509–522, 2002.

[2] C. Carson, S. Belongie, H. Greespan, J. Malik, "Blobworld : image segmentation using expectation-maximization and its application to image querying", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24(8), pp. 1026–1038, 2002.

[3] D. Comaniciu, P. Meer, "Robust Analysis of Feature Spaces : Color Image Segmentation", In Proc. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 750–757, 1997.

[4] R. Fergus, P. Perona, A. Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning", In Proc. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 264–271, 2003.

[5] M. Flickner, H. Sawney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D.P. Petkovic, "Query by Image and Video Content : the QBIC System", *IEEE Special Issue on Content-Based Picture Retrieval System*, Vol. 28(9), pp. 23–32, 1995.

[6] W.S. Geisler, B.J. Super, "Perceptual organization of two-dimensional patterns", *Psychological Review*, Vol. 107(4), pp. 677–708, 2000.

[7] K. Idrissi, G. Lavoue, J. Ricard et al. "Object of Interest based visual navigation, retrieval and semantic content identification system", *Computer Vision and Image Understanding*, Vol. 94(1-3), pp. 271-294, 2004.

[8] W.Y. Kim, Y.S. Kim, "A new region-based shape descriptor", *Technical report*, 1999.

[9] D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic, 1985.

[10] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, Vol. 60(2), pp. 91–110, 2004.

[11] F. Mokhtarian, A.K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curve", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 789–805, 1992.

[12] H. Schneiderman, T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars", In Proc. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[13] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[14] M.J. Swain, D.H. Ballard, "Color indexing", *International Journal of Computer Vision*, Vol. 7(1), pp. 11–32, 1991.

[15] J.Z. Wang, J. Lie, G. Wiederhold, "SIMPLIcity : Semantics-Sensitive Integrated Matching for Picture Libraries", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23(9), pp. 947–963, 2001.