Indexation par la forme
Utilisant les M-tree

Saïd Mahmoudi , Mohamed Daoudi
Equipe MIIRE
ENIC Telecom Lille 1
Cité Scientifique, Rue Guglielmo MARCONI
59658 Villeneuve d'Ascq France
(mahmoudi,daoudi)@enic.fr

Résumé
Dans ce travail nous avons introduit une nouvelle méthode d'indexation d'images par la forme. Nous avons proposé d'utiliser un index basé sur le CSS «Curvature Scale Space » organisé autour d'une structure d'arbre dite M-Tree, qui est totalement paramétrisée par une fonction de distance et qui permet aussi en sauvegardant les distances intermédiaires d'améliorer considérablement le temps de calcul.

Mots Clef
Indexation, CSS, M-Tree.

1. Introduction

La recherche de la similarité entre formes pose des problèmes de nature différente. Un concept de formes n'est pas présent dans l'esprit de la personne quand il émet une requête.
La représentation de forme basée sur des caractéristiques globales telles que la superficie, l'excentricité, l'orientation des axes des moments a été employé dans QBIC [5]. Ce type de représentation est invariant par rapport à certains groupes de transformations géométriques. Cependant, Malgré les bons résultats obtenus, cette approche n'utilise aucun structure arborescente.
Afin de modéliser la notion de similarité comme elle peut être perçue par un utilisateur, une méthode basée sur la déformation du croquis de l'utilisateur a été proposée par [1]. Dans cette méthode le croquis de l'utilisateur se déforme jusqu'à ce qu'il coïncide avec l'objet dans l'image. Malgré les bons résultats obtenus, il n y'a pas de structure d'arbre proposée.
Dans [3], les maximums de courbure Multi-échelle (CSS) ont été utilisés pour indexer les courbes. Cette approche a été reprise dans la norme Mpeg-7. Chaque maximum représente la convexité ou la concavité de la courbe. Il a été montré dans [3] et [4] que cette représentation est invariante à la rotation, à l'homothétie et elle est robuste aux déformations.
Cependant, la description par les maximum de CSS n'est pas complète. En plus, aucune structure arborescente n'a été proposée.
Dans cet article nous proposons d'indexer la forme d'un objet par toute l'information contenue dans le CSS, et de définir une mesure de similarité en utilisant des arbres. Chaque courbe est représentée par un ensemble Token figure (2). Cet ensemble de Tokens est organisée en arbre appelé M-tree [2] figure (3) .

2. L'Approche Multi-échelle

Il est naturel qu'un objet conserve sa forme lorsqu'il change de position et d'orientation.

Mathématiquement, cela se traduit par le fait que les formes constituent une classe d'équivalence par rapport au groupe des similitudes. Dans le cas de similitudes du plan, la courbure de la courbe $\gamma(x(u),y(u))$ par rapport à l'abscisse normalisée u est un invariant relatif (à un facteur d'homothétie près).
La courbure euclidienne est définie par :

$$k(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{(x'(u)^2 + y'(u)^2)^{3/2}}$$

où : $(x'(u),\ x''(u))$ et $(y'(u), y''(u))$ représentent respectivement les dérivées première et seconde de $x(u)$ et $y(u)$.
La courbure permet de définir différents points caractéristiques de la courbe (points d'inflexion, points de forte courbure, etc.).
Cependant, elle ne peut être utilisée comme telle car elle fait intervenir des dérivées d'ordre supérieur et elle est très sensible au bruit. Le caractère local de la courbure pose, de manière cruciale, le problème du choix de l'échelle d'analyse. C'est pour cela que nous nous intéressons à la version lissée de la courbure $k(u,\sigma)$
de la courbe γ par une gaussienne d'écart type σ.
Une analyse multi-échelle consiste en la génération d'une séquence $\gamma(u,\sigma)$, où chaque courbe de la séquence apparaît comme une version lissée, à l'échelle référencée par le paramètre σ.

3. Espace de courbure Multi-échelle (CSS)

Nous allons construire une représentation basée sur l'emplacement des points d'inflexion pour différentes échelles σ. Dans ce paragraphe nous nous intéresserons à la définition de la courbure aux différentes échelles. D'une manière formelle ceci correspond à l'équation suivante :

$$k(u,\sigma)=0$$

Elle représente l'espace de courbure multi-échelle de la courbe γ qu'on appellera CSS (Curvature Scale Space) [8,16].

Cette équation implicite représente les points d'inflexion de la courbe. Il faut bien noter que le nombre de points d'inflexion décroît quand σ croît. Nous avons présenté pour différents σ le passage par zéro de la courbure euclidienne.

La figure 1-b représente le CSS correspondant au contour d'une image correspondant à une vue d'un poisson, figure (1-a).

L'axe des X et l'axe des Y représentent respectivement l'abscisse curviligne normalisée et l'écart type σ. C'est cette représentation qui va être utilisée dans la suite. Les petits pics au niveau du CSS représentent le bruit dans le contour. Pour chaque σ, nous avons représenté les valeurs des différentes abscisses curvilignes correspondant aux différents passages par zéro. Sur la figure 3-b, on peut remarquer que pour σ=14, la courbure possède 4 passages par zéro. Comme on peut le constater, le nombre de point d'inflexion diminue au fur et à mesure que la courbe non convexe converge vers une courbe convexe.

Le CSS permet de décrire un contour. En plus, le CSS est un invariant par rapport au groupe de similitudes. Le CSS est stable par rapport aux erreurs dues aux bruits. De plus, il est complet [6]. Toutes ces propriétés sont importantes pour une application d'indexation.

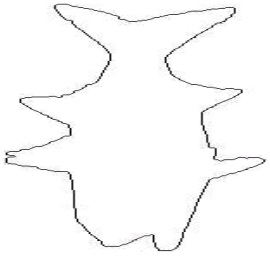


Figure (1-a) : Contour correspondant un poisson

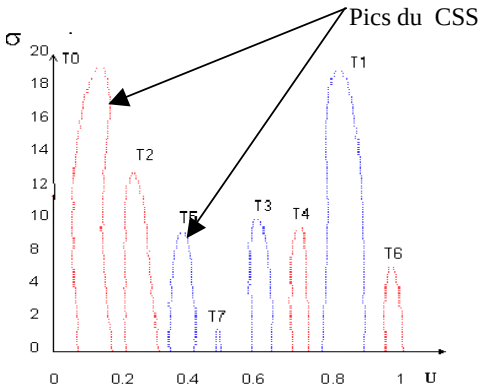


Figure (1-b) : Curvature Scale Space correspondant a la figure (1-a).

4. Indexation

Maintenant, que nous avons défini la notion de CSS, nous pouvons définir une distance entre deux CSS.

Nous utilisons la distance géodésique définie dans [7]. Etant donné deux points (s_1,σ_1) et (s_2,σ_2), avec $\sigma_1 \le \sigma_2$, leur distance D peut être définie de la façon suivante :

$$D((s_1,\sigma_1),(s_2,\sigma_2)) = \log\left(\frac{\sigma_2}{\sigma_1}\frac{\left(1+\sqrt{1-(\varphi\sigma_1)^2}\right)}{\left(1+\sqrt{1-(\varphi\sigma_1)^2}\right)-\varphi L}\right)$$

où

$$\varphi = \frac{2L}{\sqrt{\left(\sigma_1^2-\sigma_2^2\right)^2 + L^2\left(L^2+2\left(\sigma_1^2+\sigma_2^2\right)\right)}}$$

et $L = |s_1 - s_2|$ le module de la différence entre les abscisses curvilignes.

Chaque pic est défini par une ensemble de couples :

$\{(S_1,\sigma_2),(S_2,\sigma_2),\ldots,(S_n,\sigma_n)\}$

la distance entre deux pics pic_i et pic_j est défini par :

$$d(pic_i, pic_j) = \sum_{\substack{(s_p,\sigma_p)\in pic_i \\ (s_q,\sigma_q)\in pic_j}} D\left((s_p,\sigma_p),(s_q,\sigma_q)\right)$$

Mokhtarian [3] et Daoudi [4] utilisent les maximums des CSS pour indexer les formes. Cependant, en n'utilisant que les max des CSS la propriété de complétude des CSS est perdue. Les auteurs [3,4] ne proposent aucun schéma d'indexation.

Nous proposons d'effectuer le calcul de la distance entre deux pics appartenant à deux CSS différents en utilisant un ensemble de points uniformément distribués sur les deux pics, avec un pas de 10.

La distance entre deux pics pic_i et pic_j est définie par:

$$d(Pic_i, Pic_j) = \sum_{\substack{(s_p,\sigma_p)\in Pic_i \\ (s_q,\sigma_q)\in Pic_j}} D\left((s_p,\sigma_p),(s_q,\sigma_q)\right)$$

avec : $p = (n * pas)$ $\quad n = 0..\max_i / pas$

et $q = (n * pas)$ $\quad n = 0..\max_j / pas$

$\max_i$: le nombre max de points de Pic_i

$\max_j$: le nombre max de points de Pic_j

Nous proposons de structurer les pics correspondant aux CSS de chaque forme dans une structure d'arbre dites M-Tree «Metric Tree » [2,8].

Les arbres M-Tree représentent une structure de sauvegarde qui permet de ranger les couples (s_i,σ_i), qui permettent de calculer les distances entre deux pics.

Cette structure d'arbre permet de stocker les distances relatives entre les différents pics de chaque CSS afin d'éviter le calcul d'un certain nombre de distances au moment de la recherche.

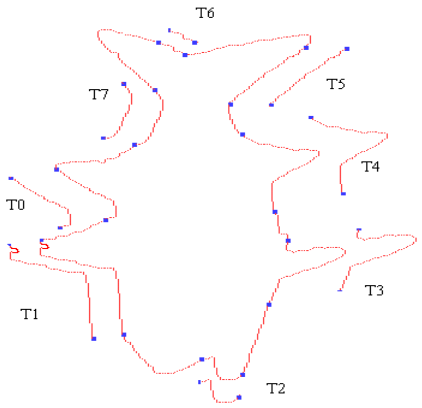


Figure 2 : Représentation du contour sous forme de Tokens. Les points qui permettent d'identifier chaque Token sont les points d'inflexion de la courbe représentant le contour.

5. Structuration des CSS en M-Tree :

Chaque courbe est représentée par un ensemble de Tokens t_i figure (2). Cet ensemble de Tokens est organisé en arbre appelé M-Tree [2] figure (3) .

Chaque forme est décomposée en un ensemble de Tokens en utilisant les CSS. Les M-Tree organisent les Tokens sous forme d'un ensemble hiérarchique de clusters.

Chaque cluster est identifié par un Objet Routeur «Object Rooting » (Le centre du cluster), et un rayon de recouvrement «Covering Radius» qui détermine la distance maximale entre l'Objet Routeur et chaque Token inclus dans le cluster figure (3).
Les entrées des feuilles de l'arbre contiennent l'ensemble des couples (S, σ) correspondants à chaque Token.
Lors de la construction de l'arbre le niveau dans lequel chaque Token doit être placé est déterminé par le procédé suivant :
Chaque nouvelle entrée (cad chaque nouveau Token à insérer) est comparée avec les Objets Routeurs dans chaque niveau , sachant qu'on commence le parcours de l'arbre à partir du «Root », nœud racine de l'arbre.
Le chemin traversant l'arbre est déterminé en sélectionnant parmis tous les Objets Routeurs celui qui minimisent l'augmentation du rayon de Recouvrement (Covering Radius) en ajoutant l'entrée en question au cluster associé à cet Objet Routeur.
Arrivant au niveau feuille s'il y a dépassement de capacité du nœud feuille une méthode de décomposition «Split» est appliqué sur ce nœud, le Split peut être propagé à partir du nœud feuille jusqu'à la racine de l'arbre (Root).

Plusieurs stratégies de Split peuvent être utilisées. On a considéré les stratégies mM-Rad (minimum of maximum radius) et MUB-Dist (maximum of lower distance), la différence réside dans le choix des Objets Routeurs. En effet, à l'occurrence d'un split, la méthode mM-Rad choisie comme Objet Routeur la paire d'entrées dont la somme des rayons de recouvrements est minimale, tant dis que la stratégie MUB-Dist choisie les deux entrées qui ont la plus grande distance comme Objet Routeur. La figure(3) montre l'organisation des Tokens de la figure(2) dans un arbre M-Tree. La taille des clusters est égale à 3.

Chaque entrée d'un nœud a le format suivant :

$$Entrée\ (t_i) = [t_i, ptr(\mathrm{T}(t_i)), r(t_i), d(t_i, \mathrm{P}(t_i))]$$

Tel que :

t_i représente l'index du Token si le nœud est une feuille, sinon il représente l'identifiant du Token

Exemple:

pour le Token t_0 :

si on est dans un nœud feuille:

$t_0 = ((s_1, \sigma_1), (s_2, \sigma_2), \ldots, (s_n, \sigma_n))$.

Sinon $t_0 = 0$, (identifiant du Token t_0).

$ptr(\mathrm{T}(t_i))$: est un pointeur vers l'adresse du sous-arbre $\mathrm{T}(t_i)$ qui inclut toutes les entrées t_j dont la distance avec t_i est inférieure au rayon de recouvrement de t_i : $r(t_i)$ donc :

$$d(t_i, t_j) \le r(t_i) \quad \forall \quad t_j \in \mathrm{T}(t_i)$$

Dans un nœud feuille $ptr(.)$ est remplacé par l'identifiant de l'image à qui appartient le Token.

$d(t_i, \mathrm{P}(t_i))$ représente la distance entre l'entrée t_i du nœud et son père dans l'arbre $\mathrm{P}(t_i)$.

La distance $d(t_i, \mathrm{P}(t_i))$ et $r(t_i)$ sont pré-calculés dans l'arbre, donc le nombre d'accès aux nœuds et le nombre de distances calculées sera réduit au moment de la recherche.

Pour un Token requête t_q et un rayon de recouvrement *r*, une technique de recherche utilisant *r* sélectionne tous les Tokens de la base telle que :

$$d(t_i, t_q) \le r$$

La distance utilisée est une métrique, donc on peut utiliser l'inégalité triangulaire pour ne pas parcourir des parties de l'arbre au moment de la recherche.
Cependant, suivant l'inégalité triangulaire, on peut facilement démontrer les deux propriétés suivantes : [2]

Propriété -1-

$$si \quad d(t_i, t_q) > r + r(t_i)$$

$$alors \quad d(t_j, t_q) > r \quad \forall \quad t_j \in \mathrm{T}(t_i)$$

Propriété -2-

$$si \quad \left| d(\mathrm{P}(t_i), t_q) - d(t_i, \mathrm{P}(t_i)) \right| > r + r(t_i)$$

$$alors \quad d(t_i, t_q) > r + r(t_i)$$

La recherche dans l'arbre est descendante.
Donc pour une entrée t_r dans un niveau donné de l'arbre: $d(\mathrm{P}(t_r), t_q)$ est calculée en raison du parcours descendent de l'arbre.

$d(t_r, P(t_r))$ est préalablement stockée dans le nœud correspondant à t_r.

Si la condition de la propriété 2 est vérifiée, on pourra éviter le parcours de $T(t_r)$, sous arbre de t_r, pendant la recherche.

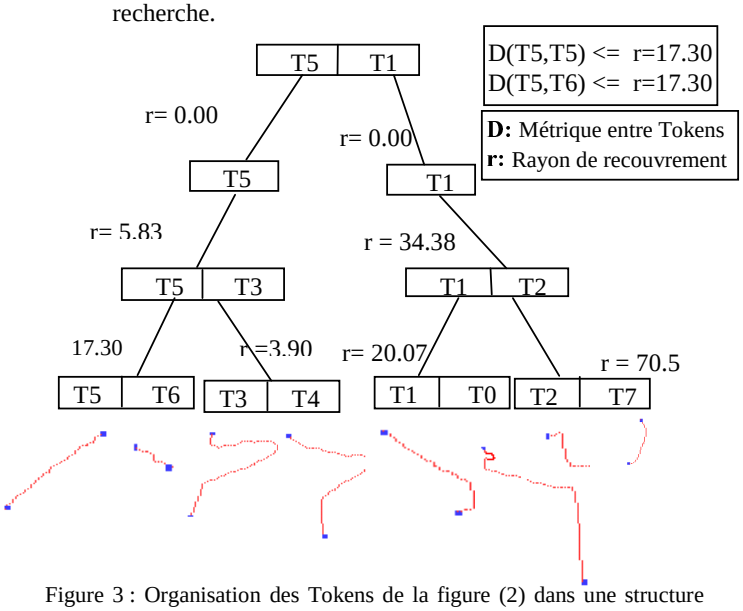
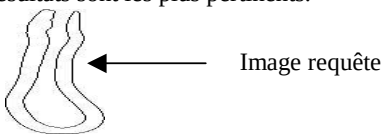


Figure 3 : Organisation des Tokens de la figure (2) dans une structure M-Tree en utilisant une stratégie de split mM_rad, la taille des clusters est supposé égale à 3.

6. Résultats expérimentaux

Nous avons testé notre application en utilisant une base de poissons composée de 1000 images. Dans la phase indexation, pour chaque image est associé un index composé d'un ensemble de points distribués d'une façon uniforme dans chaque pic du CSS. La phase finale d'indexation consiste à la construction de l'arbre M-Tree correspondant à cet index.

Les courbes rappel/précision montrent que les premiers résultats sont les plus pertinents.



Résultats de la recherche:

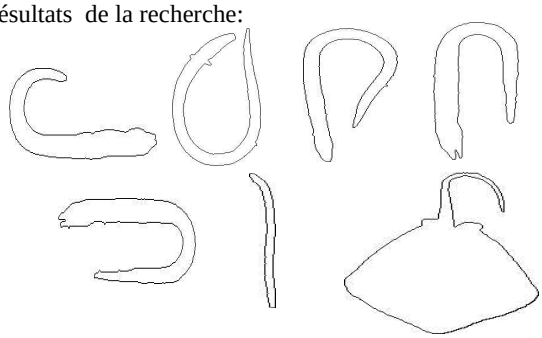


Figure 4 : Résultats expérimentaux

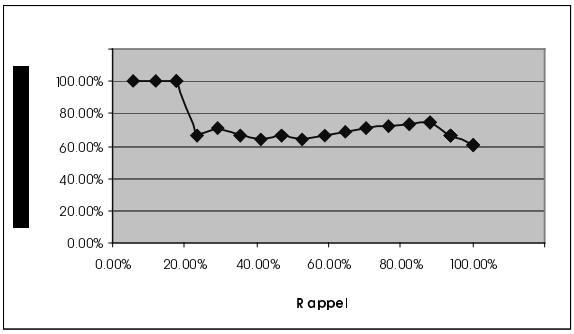


Figure 5 : Courbes rappel/précision (1000 images)

9. Conclusion :

Dans ce papier nous avons proposé une méthode pour l'indexation par la forme des images, cette méthode est basée sur les étapes suivantes :

1. Caractérisation des images par des CSS, ces CSS sont calculés à partir des contours.
2. Organisation des CSS sous forme d'une structure d'arbre dite M-Tree.
3. Comparaison des Tokens correspondants aux arbres M-Tree lors de la recherche.

Les résultats obtenus figure (4), sont encourageants et montrent qu'on peut aussi répondre à une requête partielle en utilisant notre méthode, d'où l'intérêt de notre approche.

Bibliographies

[1] A.Del Bimbo, P. Pala, and S. Santini, "Visual Image Retrieval by elastic Deformation of Objects Shapes", Proc. IEEE VL'94, Int'l Symp. Visual languages, St Louis, Mo, Oct. 1994.

[2] P. Ciaccia, . Patella, and P. Zezula "Indexing metric spaces with M-tree" in SEBD'97, 1997, pp. 67-86.

[3] Mokhtarian, F. Abbasi S. and Kittler J. (1996) : Robust and Efficient Shape Indexing through Curvature Scale Space Proceedings of the sixth British Machine Vision Conference, BMVC'96. Edinburgh, 10-12 September 1996, pp 53-62.

[4] M. Daoudi, S. Matusiak, "Visual Image Retrieval by Multiscale Description of User Sketches" Journal of Visual Computing and languages, ,, Vol. 11, 287-301, special issue on image database visual querying and retrieval, 2000.

[5] C. Faloutous et al., "Efficient and Effective Querying by Image Content" Journal of Intelligent Information Systems 3, 231-262 (1994).

[6] F. Mokhtarian and Alan K. Mackworth, " A Theory of Multiscale, Curvature-Based Shap Representation for Planar Curves" IEEE PAMI, Vol. 14, pp. 789-805, August 1992.

[7] D. H. Eberly, "Geometric Methods for analysis of Ridges in N-Dimensional Images" PhD Thesis, Univesity of North Carolina at Chapel Hill, 1994.

[8] A.Del Bimbo, P. Pala, "Retrieval by Shape Similarity with Perceptual Distance and Effective indexing", IEEE Transaction On Multimedia, 2000. Vol. 2, N°.4, December 2000.