Indexation d'images par Fractale/Viterbi

Mathieu Vissac[†], Jean–Luc Dugelay[†] et Kenneth Rose[††]
[†]Institut EURECOM – Sophia-Antipolis
{vissac, dugelay}@eurecom.fr
http://www.eurecom.fr/~image
[††]University of California – Santa Barbara
rose@ece.ucsb.edu
http://www.ece.ucsb.edu/

1 Introduction

Les bases de données d'images représentant des volumes d'informations de plus en plus considérables, l'indexation automatique basée sur le contenu est devenue un élément primordial pour gérer ces masses de données.
De nombreuses méthodes ont déjà été développées travaillant soit dans le domaine spatial, comme l'indexation par histogrammes de couleur [1], par contours [2], par textures, ou par classification d'images [3], soit dans un domaine transformé comme les codes IFS en compression fractale [4, 5]. Des produits existent déjà, et des démonstrations de ceux-ci sont visibles sur internet. Ces techniques d'indexation, malgré une vision partielle de la base de travail, montrent assez rapidement leurs limites et n'arrivent pas à modéliser correctement la notion humaine de similarité. Certaines accordent une trop grande souplesse dans le choix des images similaires (indexation par la couleur par exemple), alors que d'autres, à l'inverse, sont trop strictes, n'accordant pas assez de libertés pour le choix des transformées autorisées.

Nous proposons une nouvelle méthode d'indexation basée sur le principe de recherche par l'exemple [6] : une image est choisie comme étant représentative des images recherchées. Le programme doit alors trouver les images de la base les plus proches de l'image sélectionnée (image exemple).
Notre méthode se base sur la notion de similarité incluse dans la compression fractale [7]. Cependant il s'agit d'une recherche de similarités inter-images et non intra-image. De plus certaines contraintes, comme la notion de contractivité ou la compacité du dictionnaire, liées à l'aspect "compression", ne sont plus prises en compte dans cette version repensée pour l'indexation. Il s'agit, en effet, d'une comparaison bloc à bloc (ou plus généralement primitive à primitive) de deux images autorisant diverses transformées géométriques (rotations et symétries) ainsi que de faibles ajustements photométriques.
Pour évaluer la ressemblance entre deux images, nous utilisons la notion de similarité locale issue du codage fractale avec une gestion globale via l'algorithme de Viterbi. Ce dernier nous permet de prendre en compte différentes contraintes (continuité de la transformée, continuité spatiale, ...) tout en nous certifiant le choix optimal des paramètres (choix du bloc test, choix de la transformée, ...) et l'obtention du meilleur résultat.

Notre méthode peut donc se décomposer en deux parties :

Similarité locale : Cette étape permet, par sa liberté et sa souplesse, d'englober la notion humaine, assez floue, de similarité.

Cohérence globale : Cette seconde étape permet d'ordonner les différentes similarités locales en un tout. Sous la contrainte que cette liste de transformations locales ait, au sens visuel, une cohérence globale.

Dans la section 2, nous présentons comment nous recherchons toutes les similarités locales possibles en parcourant l'image exemple (section 2.1), puis comment nous transformons les erreurs d'appariement calculées dans la section 2.2 en probabilités (section 2.3). Dans la section 3 nous appliquons l'algorithme de Viterbi (section 3.1) afin de de relier les différents appariements locaux en respectant des contraintes de cohérence globale (section 3.3).

2 Similarité locale

Le but est, ici, de trouver toutes les similarités locales entre deux images. Nous travaillons sur des morceaux d'images (blocs) afin de faciliter la recherche et de permettre de petites modifications locales (ex : modification de l'éclairage de l'objet).

2.1 Parcours de l'image

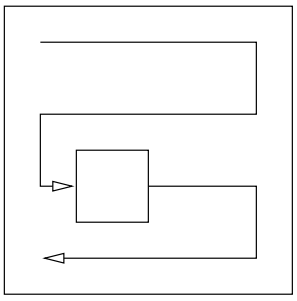


FIG. 1 – *Parcours en serpent*

Par analogie avec la compression fractale, l'image exemple est partitionnée en un ensemble de blocs. Ces blocs sont parcourus dans un ordre défini afin de nous donner une relation de *suite* entre les blocs. Il existe plusieurs parcours possibles qui permettent de définir une notion de voisinage. Nous avons opté pour le parcours en serpent qui répond bien à nos exigences en assurant que chaque bloc soit voisin avec le bloc précédent.

2.2 Recherche de similarités

Ensuite, pour chaque bloc de l'image source, nous recherchons dans toute l'image testée, les blocs présentant la plus grande similitude. Le calcul de la similitude est fait par les moindres carrés après ajustement des paramètres de transformation.

Nous avons les paramètres géométriques $(a_i, b_i, c_i, d_i, e_i, f_i)$ dérivés de la compression fractale.
Nous obtenons, pour des images binaires, l'équation suivante :

$$\tau_i . \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ 0 \end{pmatrix}$$

Les résultats obtenus sont des erreurs d'appariement E calculées avec l'équation suivante :

$$E(n,i,j) = \sqrt{\frac{\sum_{pixels}(S_n - \tau_i . D_j)^2}{\#\text{ pixels}}}$$

ou S_n est le n-ième bloc de l'image exemple
et D_j est le j-ième bloc de l'image testée

Actuellement, la translation définie par e_i et f_i n'est pas prise en compte lors des calculs des états (voir section 3.2). En fait, pour chaque transformée géométrique i, nous ne conservons que le meilleur score d'appariement Δ:

$$\Delta(n,i) = \min_j(E(n,i,j)$$

2.3 Calcul des probabilités

Afin que les calculs effectués dans la section 3 soient homogènes, nous transformons cette erreur d'appariement en une probabilité que nous appellerons *probabilité simple*.
La conversion peut se faire de différentes manières sachant que plus l'erreur est grande, plus la probabilité simple est faible.
Nous avons implanté une première fonction simple ($f(x) = 1 - \frac{x}{255}$) de conversion afin de réaliser les premières simulations (voir figure 2). Les résultats montrés dans ce papier ont été obtenu avec cette fonction. Nous pensons, cependant, changer pour une fonction gaussienne ($f(x) = e^{-\frac{x}{A}^2}$) afin de pouvoir régler, grâce à la variance A, un seuil de probabilité délimitant plus nettement l'intervalle d'erreur correspondant à deux blocs similaires de celui correspondant à deux blocs différents. Nous obtenons donc une probabilité simple définie par l'équation suivante :

$$O_n(i) = f(\Delta(n,i))$$

3 Cohérence globale

L'utilisation de l'algorithme de Viterbi dans l'indexation d'image permet d'imposer une cohérence globale dans un ensemble de similarités locales. Les modèles de Markov et l'algorithme de Viterbi sont utilisés dans d'autres secteurs de recherches comme la reconnaissance vocale [8] ou la classification d'images [9].

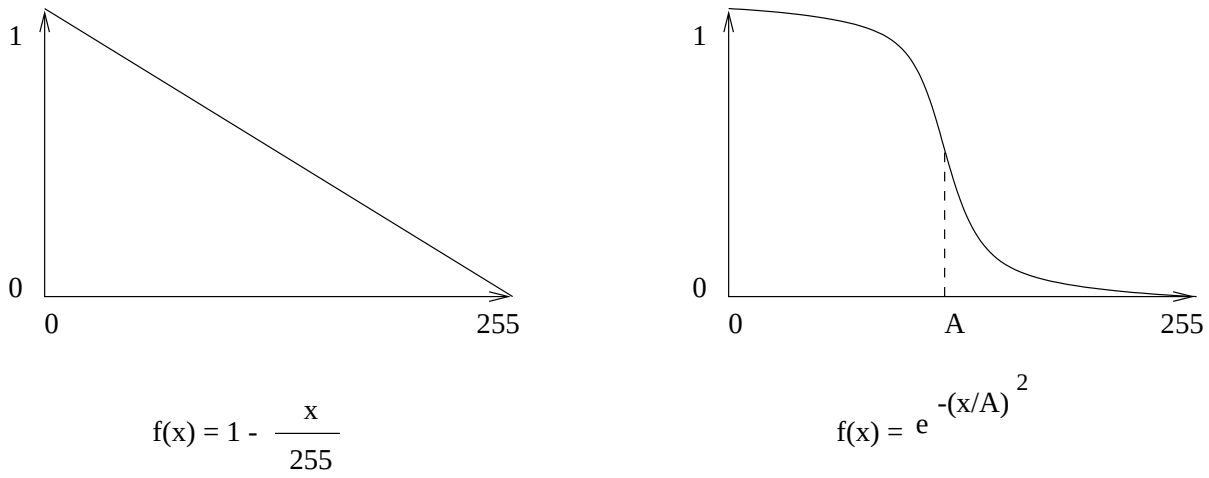
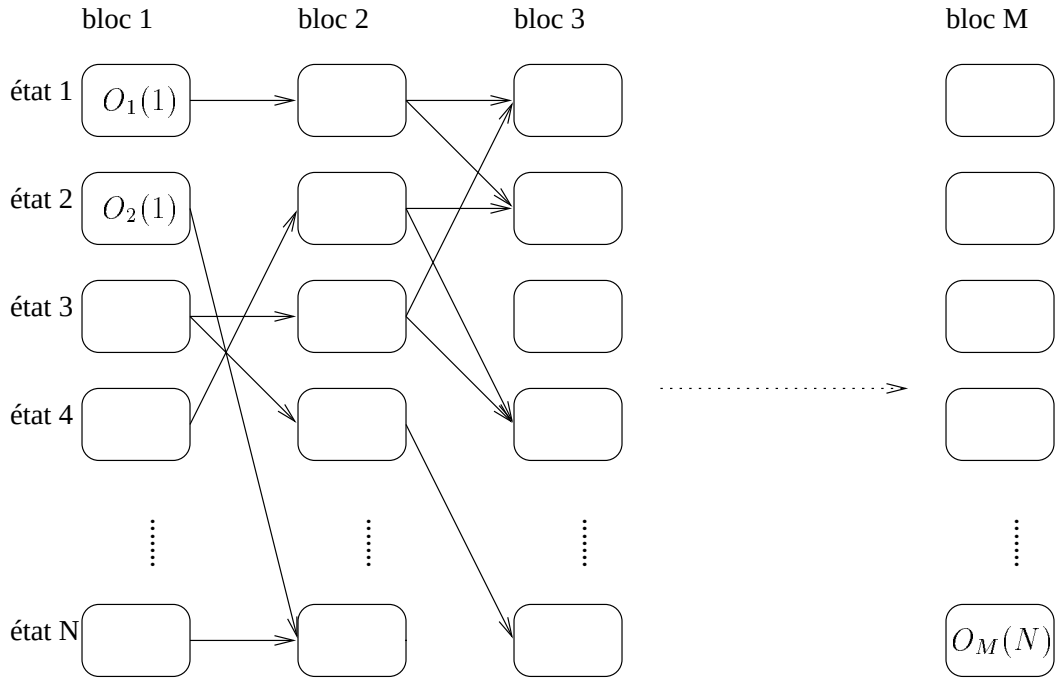


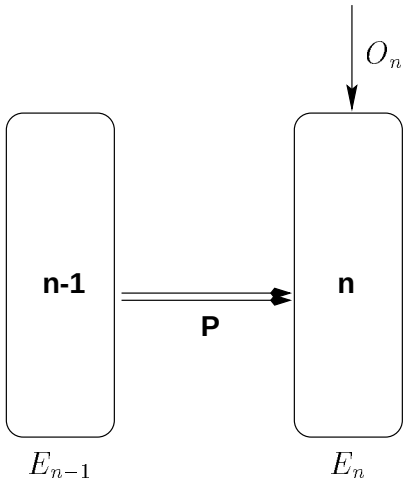
FIG. 2 – *Conversion des erreurs d'appariement en probabilité*

3.1 L'algorithme de Viterbi

L'algorithme de Viterbi permet de trouver le meilleur chemin dans un treillis d'états. Ceci se fait de manière itérative en appliquant des probabilités de changement d'états.
Nous allons détailler les modifications que nous avons appliqué à cet algorithme pour l'appliquer à notre problème. Pour les bases de l'algorithme de Viterbi, le lecteur est invité à consulter les papiers [10, 11].



La probabilité totale d'un état pour un bloc donné est donc calculé de la manière suivante :



E_{n-1} est l'ensemble des probabilités du bloc $n-1$ pour tous les états

O_n est l'ensemble des probabilités simples pour chaque états du bloc n $(O_n(1)...O_n(N))$

P représente les probabilités de changement d'états

N est le nombre total d'états

On obtient donc : $E_n(j) = \max_i(E_{n-1}(i) \times P(j|i) \times O_n(j))$

L'ordre de parcours défini dans la section 2.1, nous permet d'utiliser l'algorithme de Viterbi en une version 1D, puisque nous avons défini un ordre de parcours ou chaque bloc a un côté en commun avec son prédécesseur.
Le meilleur chemin est donc calculé comme étant un compromis entre la somme cumulée des erreurs d'appariement et les pénalités dues à des ruptures dans la continuité de la transformée.

3.2 Les états

Les fonctions τ_i, qui relient les blocs deux à deux, sont des transformations traduisant la similarité locale. Les états permettent de définir les caractéristiques autorisées de ces transformations.

Nous avons, pour les tests préliminaires, uniquement travaillé sur des images binaires. Nous avons donc implanter qu'un seul groupe d'états correspondant aux transformées géométriques locales pouvant être appliquées à chaque bloc. Ces transformées sont celles classiquement utilisées en codage fractale :

0	Identité	4	Symétrie 2ème diag.
1	Symétrie Horizontale	5	Rotation de $\frac{\pi}{2}$
2	Symétrie Verticale	6	Rotation de π
3	Symétrie 1ère diag.	7	Rotation de $\frac{3\pi}{2}$

3.3 Les pénalités

Après avoir défini, pour chaque bloc de l'image exemple, les blocs correspondant à chaque état, il nous faut choisir le *meilleur chemin* dans le treillis. Pour cela, nous définissons un ensemble probabilités correspondant à toutes les transitions possibles d'état.
Afin de garantir une cohérence globale dans la similarité, ces probabilités limitent

les changement d'états entre deux blocs successifs.
En effet, soit I_t l'image testée, similaire à l'image source I_s modulo une transformation $\tau = \tau_g$, ou τ_g est la transformée géométrique. Chaque bloc de I_s correspond à un bloc i de I_t modulo une transformation τ_i.
Vu que I_t est similaire à I_s modulo τ, chaque bloc de I_t doit subir la même transformation τ pour correspondre à un bloc de I_s : $\forall i \tau_i = \tau$.
Nous utilisons donc des probabilités de changement d'états qui favorisent une certaine stabilité dans le choix des transformées τ_i.

4 Résultats préliminaires

Les requêtes suivantes ont été effectué sur une partie (de l'image indicée 3000 à l'image indicée 3084) de la base de logos noir et blanc de MPEG-7 (*Trademark images captured by a scanner*, CD-Rom num. 10).

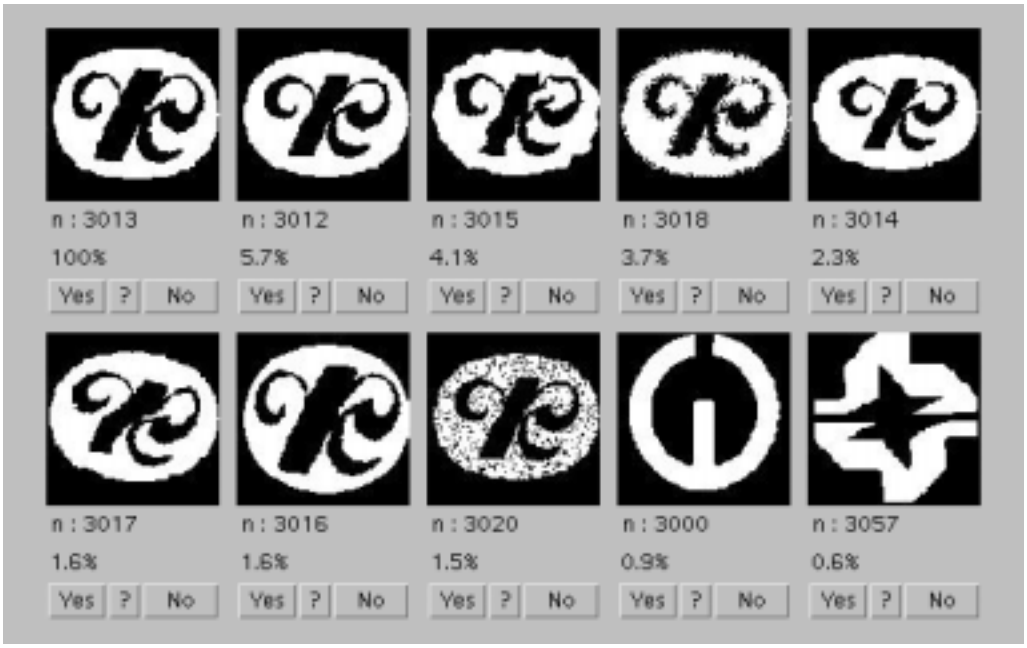


FIG. 3 – *Image exemple : 3013 (efficacité = 89%)*

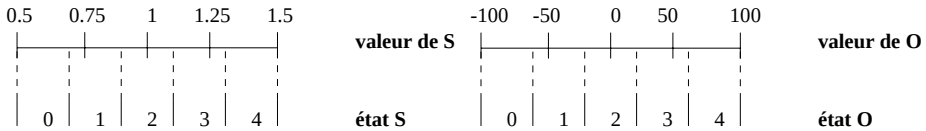
L'efficacité indiquée pour chaque résultat (voir figures 3 et 4) est calculée comme étant le nombre d'images similaires retournées jusqu'à la première non-similaire (S_{ret}) divisé par le minimum entre le nombre total d'images similaires dans la base (S_{tot}) et le nombre d'images demandé dans la requête (N_{ret}).

$$\text{efficacité} = \frac{S_{ret}}{\min(S_{tot}, N_{ret})} \times 100\%$$

FIG. 4 – *Image exemple : 3031 (efficacité = 100%)*

5 Remarques concluantes

Notre algorithme possède un fort potentiel d'adaptation à de nouvelles requêtes. En effet, l'utilisation de l'algorithme de Viterbi autorise facilement l'ajout de nouveaux états pour caractériser de nouvelles transformations comme un ajustement photométrique ou un changement d'échelle. Afin de pouvoir traiter des images en niveaux de gris, nous pouvons, par exemple, concevoir un groupe d'états caractérisant des opérations sur les valeurs de luminance. Ce groupe comprendrai deux sous-groupes, un pour le "*scale*" de luminance et l'autre pour "l'*offset*".



De plus, l'introduction d'un groupe d'états adaptés caractérisant les positions relatives des blocs correspondants est en cours afin d'éviter un éventuel effet puzzle de l'image. Ce dernier ne faisant pas toujours partie des transformations autorisées. L'implantation de ce groupe d'états est actuellement en cours.

Pour conclure, bien qu'en étant au début des recherches, les premiers résultats obtenus montrent que l'utilisation de similarités locales couplé avec l'algorithme de Viterbi est une approche prometteuse avec un grand potentiel d'évolution.

Références

[1] T. Randen and J.H. Husoy. Image content search by color and texture properties. In *ICIP-97*, volume 1, pages 580–583, 1997.

[2] C. Nastar. The image shape spectrum for image retrieval. Technical Report 3206, INRIA, July 1997.

[3] H. Yu and W. Wolf. A hierarchical, multi-resolution method for dictionary-driven content-based image retrieval. In *ICIP-97*, volume 2, pages 823–826, 1997.

[4] J.M. Marie-Julie and H. Essafi. Image database indexing and retrieval using the fractal transform. In *ECMAST'97*, pages 169–182, 1997.

[5] A. Zhang, B. Cheng, and R. Acharya. A fractal-based clustering approach in large visual database systems. *Multimedia Tools and Applications*, (3):225–244, 1996.

[6] C. Nastar. Indexation d'images par le contenu : un etat de l'art. In *CORESA'97*, 1997.

[7] A. E. Jacquin. Image coding based on a fractal theory of iterated contrative image transformation. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1991.

[8] L.R. Rabiner and Juang B.H. *Fundamentals of speech recognition.* Prentice-Hall, 1993.

[9] J. Li, A. Najmi, and R. Gray. Image classification by a two dimensional hidden markov model. In *ICASSP'99*, volume 6, pages 3313–3316, March 1999.

[10] M.S. Ryan and G.R. Nudd. The viterbi algorithm. Technical report, University of Warwick, England, 1993.

[11] A. J. Viterbi. Errors bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, 1967.