Digital Image Indexing and Retrieval by Content using the Fractal Transform for Multimedia Databases

Jean Michel Marie-Julie - Hassane Essafi
LETI(CEA-Technologies Avancées)
DEIN - CEA Saclay F91191 Gif sur Yvette Cedex France
E-mail: {jmmjulie, hessafi}@gaap.saclay.cea.fr

Abstract

Digital image database represent huge amount of data, automatic indexing and content base retrieval are crucial factors. Content base retrieval requests specific indexing techniques allowing to deal with efficient data representations (reduced amount of data and adapted to content base retrieval) in order to avoid prohibitive low level process time on queries.

We present a method consisting in building an "iconic" level image representation preserving the semantic, allowing to process content base retrieval and to reconstruct the image. This representation consists in a set of function parameters. It is a semantic preserving compression built by using a dedicated Fractal compress scheme. To search patterns in an huge set of images, a specific algorithm allows to use this "iconic" representation as an index. It works entirely in the Fractal transform parameter space of both image and pattern, to obtain performances compatible with an interactive search.

The research engine uses both textures and edges of the pattern. The pattern can be present in the image with different orientations and/or scales by using a multiresolution Fractal representation of the pattern.

This method allows to retrieve in 3 seconds a 64 x 64 pixels pattern in an 100 images (512 x 512 pixels) database, on a SUN Sparc 20 workstation. It can be combined with other indexing and retrieval techniques, such as textual annotation.

1. Introduction

Today technologies allow to store huge amount of document including images. The document database management needs the development of dedicated methods for the indexing and retrieval of images.

Textual annotation is a very common technique, it consists in associating to the image a text used by traditional database research algorithm. The generation of the automatic text description is hard because of some pattern recognition problems. Regarding the manual generation of the text, it is subjective and influenced by requests, the database is dedicated to a particular problem, and have to be re-annotated for each new application.

To alleviate this drawback, a solution is to use an automatic indexing technique to generate a representation, which allows to preserve the semantic of the image (spatial relations), and then to support different sort of queries. Process to obtain this description are query dependant (see *table* [1]). There are different sort of queries based on appearance, shape, colours, texture, or pattern. Correlation techniques [4] or eigen-representations (*Karhunen-Loève* expansion) [5], for instance, allows to process search based on appearance. To process queries based on shape, some methods use geometric invariants of shape moments [6][8][9] or eigen-representations [6]. Colour histograms [7][10][11] are used to process colour based queries. Texture based queries, can be processed by using a Wold decomposition [5], to characterise texture classes, for instance.

All these methods associate a parameter set to the image, adapted to the query. They can be use together to process complex queries (based on colours, shapes and textures). Some methods allow to preserve the image semantic and to reconstruct the image (eigen representations) or the texture (Wold texture decomposition).

For queries based on pattern matching, traditional indexing/retrieval methods does not allow to reduce enough the retrieval time. The correlation method, even coupled with a wavelet decomposition is not as fast as needed (on scalar workstation) for interactive search process. The method presented in this paper proposes an efficient indexing to obtain small query processing time.

Our method consists in two steps, the indexing and the retrieval. The indexing step uses a dedicated Fractal compression scheme, this offers an image representation preserving the semantic. This representation contains the image compressed data, allowing to reconstruct the image by decompressing, and a compact index used during the search process. The search engine compares this index with a multiresolution or multi-compression representation of the pattern. Furthermore, thanks to a multi-compression or multi-resolution representation of the pattern, it is possible to retrieve this pattern even if its orientation or scale in the image is different.

The advantage of the search process is that it uses directly compressed data, without decompressing them, and avoid using low-level processes, so it allows to reduce considerably the query process time (see the result section).

The indexing and retrieval scheme can be applied to any application field, it does not require any knowledge about image content (no pre-process required as background/foreground separation) and no learning step.

In a first part, the Fractal compression is exposed, then the image indexing, which uses a particular compression scheme, is described. The search is presented in a third part, followed by some results and a conclusion.

Request type	***Method***	***Indexing required***	***Low-level process required for the search***	***Reconstruction from the index***
Appearance	• Eigen representation[5]	**Yes**	-	**Yes**
Shapes	• Eigen representation[5]	**Yes**	-	**Yes**
	• Moments [6][8][9]	**Yes**	-	-
Textures	• *Wold* decomposition [5][6]	**Yes**	-	**Yes**
	• Multifractal dimension[12][13]	**Yes**	-	-
Colour	• Colour histogram [17][10][11]	**Yes**	-	-
Pattern search	• Multiresolution correlation[14][15]	**Yes**	**Yes**	**Yes**
	• Fractal compression	**Yes**	-	**Yes**

Table [1] : Request types and some examples of associated methods (non-exhaustive list).

2. Fractal compression

The basic idea of the Fractal compression [1][2][3] is that the image is the attractor of a contractive map. This map is an IFS (Iterated Function System) and can be described by its parameters set. The aim of compression is to determine these parameters and so the IFS map.

First, we consider the image *f* as a function defined on [0,1]x[0,1]=I^2 such as:

$$f: I^2 \to I$$
$$(x,y) \to z = f(x,y)$$

The image *f* is the attractor of a map *W*:

$$f \equiv |W| = \lim_{n\to\infty} W^{o n} = WoWoWo...oW$$

W is a set of n transformations.

$$w_i : B^2 \to B^2 \quad B \text{ is a compact of } I^2$$

W can be written as:

$$W(B) = \bigcup_{i=1}^{N} w_i(B) \; B \text{ is a compact of } I^2$$

The Hutchinson theorem tells us that *W* is contractive, in a space of (closed and bounded) subsets of the plane, if all transformations w_i are contractive. The transformation w_i is contractive if:

$$d\left(w\left(P_1\right), w\left(P_2\right)\right) < s \cdot d\left(P_1, P_2\right) \text{ with } s < 1 \text{ and } d \text{ a distance}$$

If the map W is contractive, it has a unique fixed point $|W|$, $|W|$ is called the attractor of W.

2.1 Application to Image - PIFS (Partitioned Iterated Function System)

Common images are not Fractal, they are not globally self-similar, but they are locally self-similar. So PIFS (*Partitioned Iterated Function System)* are introduced, they are collections of local transformations, w_i. Each transformation w_i is applied to a part of the image. A PIFS is a collection of transformations w_i, $i \in [0;N]$. w_i can be written as:

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}$$

w_i contains a geometric part (isometrie and scaling) and a massic part applied to image pixels.

s_i and o_i are respectively the contrast scaling and luminosity offset parameters. a_i, b_i, c_i, d_i parameters are used for the isometrie (rotation, reflection or scaling). e_i and f_i define the translation.

The image f is decomposed it into N non-overlapping *ranges* blocks R_i and decomposed into M overlapping *domains* D_j, as:

$\cup R_i = I$ and $R_i \cap R_k = \varnothing$ with $i \in [0;N]$, $k \in [0;N]$ and $i \neq k$

Each transformation w_i transforms a *domain* D_j into a *range* R_i.

$$w_i\left(D_i\right) = R_i$$

2.2 Image encoding

To encode an image f, we need to find a transformation set w_1, w_2, ..., w_N with $W = \bigcup_{i=1}^{N} w_i$ and $f=|W|$. In other words, f is the fixed point of W:

$$f = W(f) = w_1(f) \cup w_2(f) \cup \ldots \cup w_N(f)$$

In reality we are looking for:

$$f \approx f' = W(f') \approx W(f) = w_0(f) \cup \ldots \cup w_N(f)$$

To determine the w_i transformations, we calculate for each range R_i, the domain D_i which is the most similar, considering an isometrie and contrast scaling and luminosity offset.

The similarity is measured by the root mean square distance, to adjust s_i and o_i.

$$R = \sum_{i=1}^{p} (s \cdot a_i + o - b_i)^2$$

a_I and b_i are respectively the grey levels of a *domain block* (after isometrie) and of a *range block*.

The w_i can be stored as vectors W_i:

$$W_i = \left(\gamma_i \quad x_i \quad y_i \quad tx_i \quad ty_i \quad s_i \quad o_i\right)$$

- γi Isometrie code;
- γ_i isometrie code;
- x_i, y_i domain position;
- tx_i, ty_i, w_i translation coordinates of w_i;
- s_i, o_i contrast scaling and luminosity offset

2.3 Image decoding

To decompress the image, the fixed point or attractor is calculated by iterating the PIFS (see *figure* [12]).

3. Indexing

The Fractal image compression gives a set of parameter vectors, describing a PIFS (Partitioned Iterated Functions System). The idea is to index the image with the PIFS, this index can be used for content based retrieval.

To achieve that, we have to adapt the compression scheme. Then, we apply a filtering to the index to make it as compact as possible, to reduce the processing time.

The indexing is fully integrated to the compression process, they can be done simultaneously.

3.1 Adapted compression scheme

3.1.1 Reduction of the domain search space

The modification of the compression scheme is motivated by the fact that the Fractal compression is a global process. This process can be seen as a vector quantization, where the dictionary contains all the domain blocks. For each range block, we search which domain block is the most similar. This implies that the encoding of a given image part, if we consider a global dictionary (all the domain blocks) is different compared to the encoding of the same part, if the dictionary only contains domain blocks of this part (see *figure* [1]).

In order to alleviate this drawback, the research area of each range block is reduced to its neighbourhood (domain blocks belonging to a restricted search space). The size of this neighbourhood is an important parameter, the pattern size will not be able be smaller than it.

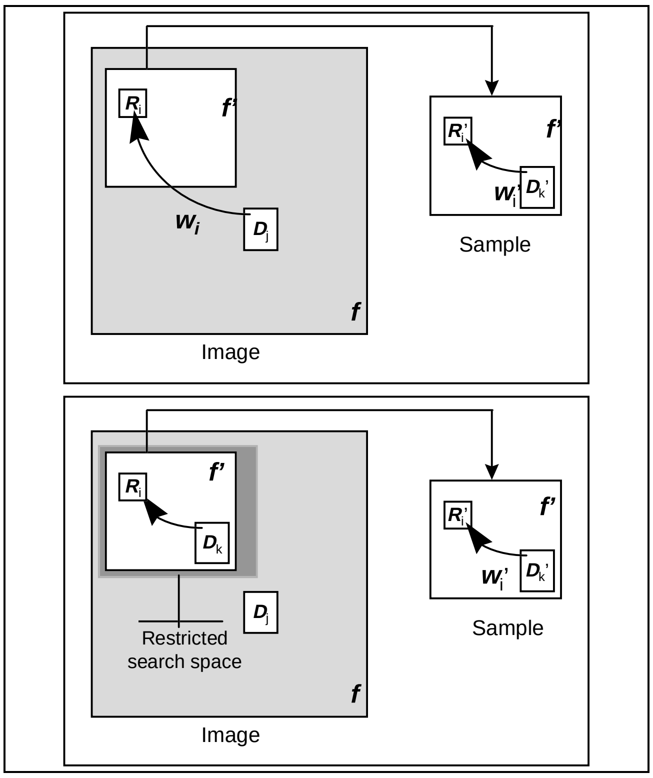


Figure [1] : In the first case (top), w_i associates D_l to R_i, if the part f' is compressed independently, the transformation $w_{i'}$ associates D_k' to R_i. The transformations w_i and w_i' are different. In the second case, a restricted search space is used, the transformations w_i' and w_i are identical.

3.1.2 Choice of the domain blocks

Generally, during the compression process, a domain partition is built. But nothing assumes that the image and a part of this image independently considered, have the same range partition and domain blocks (see *figure* [2]).

During the request processing, this implies to compress several times the pattern with different partition offsets. For instance, if the image is partitioned with 4x4 pixels range blocks and with 8x8 pixels domain blocks, we have to compress 64 times the pattern by translating the partition on both directions, to cover all possibilities.

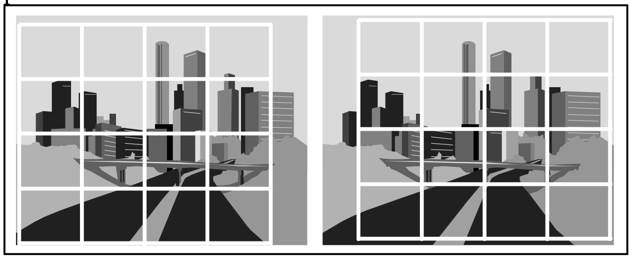


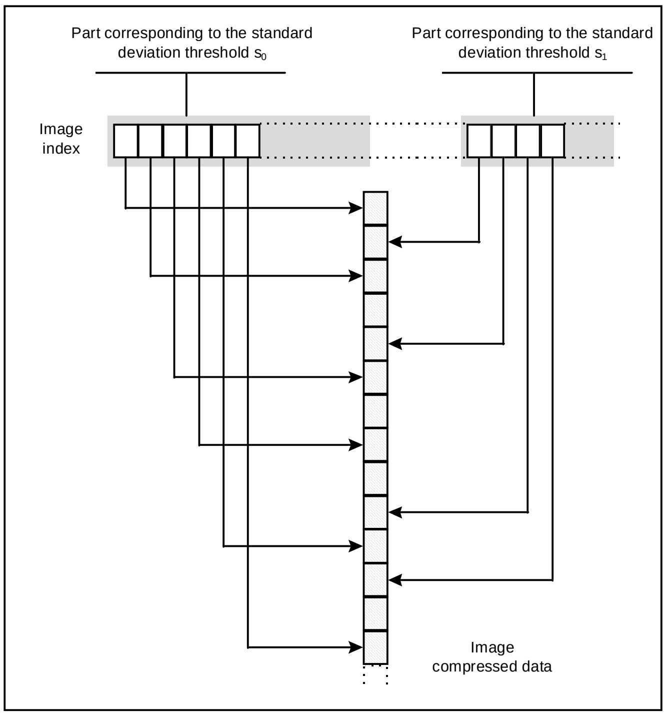
Figure *[2] : Two different partitions of a image part.*

The solution is to consider all domain blocks of constant size on the image.

3.1.3 Index filtering

To make the index more compact, PIFS are classified. Given a class, C_k of PIFS, w_i such as $w_i(D_i)=R_i$, belongs to C_k if $q(R_i)>s_k$. The q() function allows to measure the information quantity available in the range block R_i, q() could be the standard deviation of the pixel distribution of R_i. The s_k values are the index thresholds.

We can organise the index as shown in *figure* [3].



Part corresponding to the standard deviation threshold s_0
Part corresponding to the standard deviation threshold s_1
Image index
Image compressed data

Figure *[3] : The index is partitioned into classes according to the standard deviation.*

3.2 Multiresolution indexing

The multiresolution indexing allows to find a pattern even if its size is different in the candidate image. To achieve that, we consider that PIFS describe relationships between parts of the image (between range blocks and domain blocks). These relationships can be represented as graphs for different resolutions (different levels) (see *figure* [4]). The G_n graph corresponds to the level n.

At level 0, the G_0 graph describes relationships between range blocks and domain blocks. A graph node is a range block or a domain block, a link between two nodes denotes a PIFS parameter vector W_i, which transforms the domain block D_j into the range block R_i.

At level 1, nodes are set of range blocks or a domain block of level 0. We expose further how to build the links.

At level n, nodes are set of nodes of level n-1.

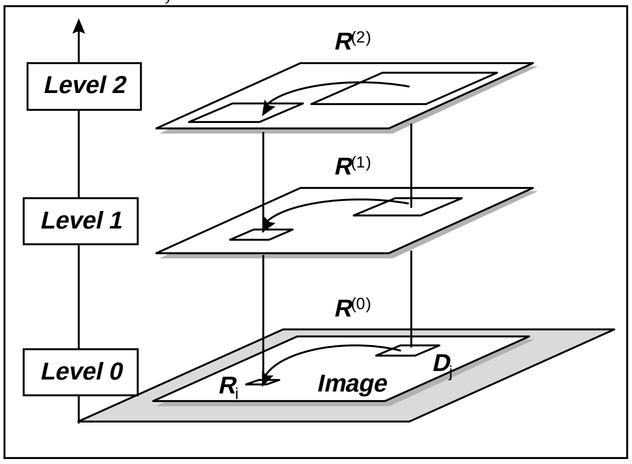


Figure *[4] : Multiresolution representation of an image.*

3.2.1 Building of the graph set

The building of the graph set is an iterative process, we start by creating the G_0 graph directly from the PIFS.

We write as $.^{(n)}$, the objects belonging to level n. For instance, $R^{(n)}$ is a map defined at level n and $p^{(n)}$ is a part belonging to level *n*.

We consider $P^{(n)}$ as the set of nodes belonging to level n. For instance, for the level 0, it is the set containing range and domain blocks.

The graph $G^{(n)}$ represents the map $R^{(n)}$:

$$R^{(n)}(p^{(n)},q^{(n)}) \text{ with } p^{(n)},q^{(n)} \in P^{(n)}$$

$$\text{if } \exists R^{(n-1)} \text{ such as } R^{(n-1)}(D_k^{(n-1)},D_m^{(n-1)})$$

$$\text{with } D_k^{(n-1)} \subset p \text{ and } D_m^{(n-1)} \subset q$$

In other words, a part $p^{(n)}$ is related to a part $q^{(n)}$ if one or more part of $p^{(n)}$ is related to one or more part of $q^{(n)}$ (parts of $p^{(n)}$ belong to level n-1) (see *figure* [5]).

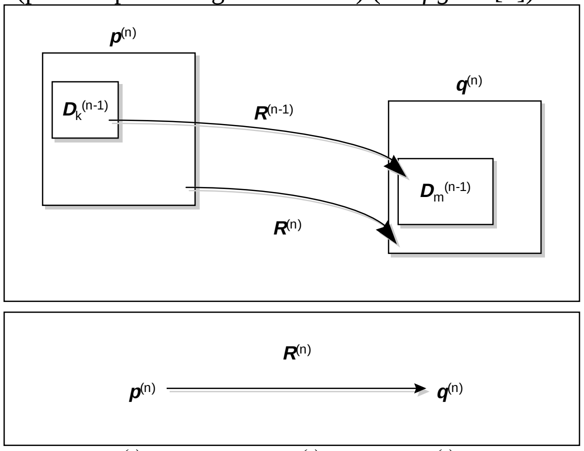


Figure *[5]: $p^{(n)}$ is related to $q^{(n)}$ because $p^{(n)}$ is containing part(s) related to part(s) of $q^{(n)}$.*

The map $R^{(n)}$ owns three attributes, which are the average of contrast scaling ($\bar{s}$) and the luminosity offset values ($\bar{o}$) of the PIFS and the number of parts concerned by the relation (*v*).

We note :

$$R^{(n)}(p^{(n)}) = \left\{ q^{(n)} \middle| R^{(n)}(p^{(n)},q^{(n)}) \text{ is defined} \right\}$$

as being the set of part related to $p^{(n)}$. Then we can write :

$$\begin{cases} \bar{s} = \dfrac{1}{v} \sum\limits_{r_k \in p_i^{(1)}} s_k \\ \bar{o} = \dfrac{1}{v} \sum\limits_{r_k \in p_i^{(1)}} o_k \\ v = card(R^{(n)}(p^{(n)})) \end{cases}$$

with s_k and o_k contrast scaling and luminosity offset values of $R^{(n-1)}(p^{(n)})$.

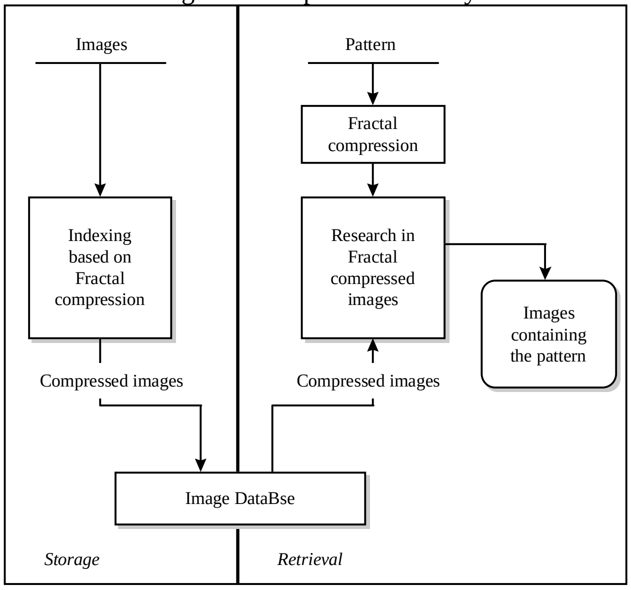
4. Content Base Retrieval

4.1 Search strategy

The search algorithm allows to retrieve an image by giving a pattern. It compares the image PIFS (index) with several PIFS of the pattern (see *figure* [6]).

To allow the retrieval of the image, for any size of the pattern, the pattern is multi-compressed or compressed with a multiresolution scheme, as described in the indexing part.

Thanks to a suited data structure, the comparison between the image and sample PIFS is very fast.



Images
Pattern
Fractal compression
Indexing based on Fractal compression
Research in Fractal compressed images
Images containing the pattern
Compressed images
Compressed images
Image DataBse
Storage
Retrieval

Figure *[6] : General synopsis.*

4.2 Pattern compression

The pattern compression contains several steps depending of the type of search. Multiresolution or Multi-compression is used to retrieve a pattern in an image with any size or orientation (see *table*[2]).

Multiresolution compression is generated by the same process as presented in the indexing part. Multi-compression consists in compressing the pattern with different size and rotations, but the compression process is time consuming. To avoid multiple compression processes, the multiresolution representation can be used.

4.2.1 Multiresolution

The multiresolution representation of the pattern includes a set of graphs at different levels. These graphs can be compared with the multiresolution index of the image, as described further.

4.2.2 Multi-compression

Multi-compression allows to cover different scales or orientations just by changing the range partition (see *figure* [7]).

We do not need to generate all the PIFS corresponding to all possible rotations. We can use some properties of the PIFS parameters to reduce the number of compression process required, as described in the next part.

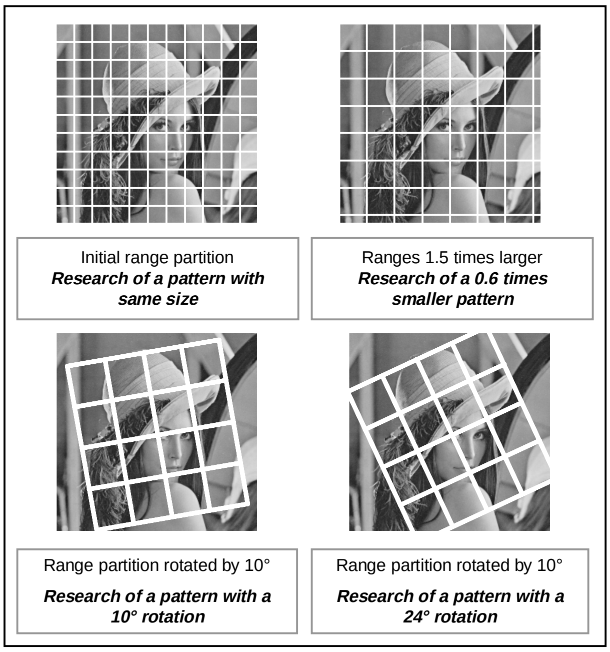

Initial range partition
Research of a pattern with same size
Ranges 1.5 times larger
Research of a 0.6 times smaller pattern
Range partition rotated by 10°
Research of a pattern with a 10° rotation
Range partition rotated by 10°
Research of a pattern with a 24° rotation

Figure *[7] : To obtain a other orientation or scale of the pattern, we just need to modify the range block and the domain block partitions.*

Compression scheme	Applied on pattern and/or image	PIFS number for the sample	Size invariance	Rotation invariance	Reflection invariance
Mono-compression	pattern and image	1	**-**	**-**	**Yes**
Multi-compression	pattern only	$1\times n_{size} \times n_{rotation}$	**Yes**	**Yes**	**Yes**
Multi-resolution	pattern and image	$1 \times n_{rotation}$	**Yes**	**Yes**	**Yes**

Table *[2]: Number of PIFS used and invariance properties of different methods for the pattern and/or image compression. The range size used for the sample is $x_{range}\, y_{range}$ pixels ,n_{size} and $n_{rotation}$ are respectively the number of different sizes and the number of different rotations for the pattern partition.*

4.2.2.1 Multi-compression and rotation

The first property concerns the different reflections (horizontal and vertical reflections, diagonal reflections). If we consider the PIFS corresponding to a given image, the PIFS of the image reflected is the reflection of the PIFS.

In other words, if I is an image and $\psi(I)=I_R$ is the reflection of I, we write H_R as the reflection applied to a PIFS (ifs_I), thus we write :

$$H_R(ifs_I) = ifs_{I_R}$$

In other words the PIFS (*ifs*) describing the rotated image (I_R) is the rotated PIFS (ifs_{I_R}) obtained with $H_R(ifs_i)$.

The transformation h() is applied to parameter vectors W_i belonging to the PIFS. We note respectively h_{RH}, h_{RV}, h_{RD1}, h_{RD2}, as the horizontal, vertical, first diagonal and second diagonal reflections. We can write for each of them :

$$h_{RH}(W_i) = h_{RH}(\gamma_i, x_i, y_i, tx_i, ty_i, s_i, o_i)$$
$$= (\gamma_i, x_i, y_i, tx_i, -ty_i, s_i, o_i)$$
$$h_{RV}(W_i) = h_{RV}(\gamma_i, x_i, y_i, tx_i, ty_i, s_i, o_i)$$
$$= (\gamma_i, x_i, y_i, -tx_i, ty_i, s_i, o_i)$$
$$h_{RD1}(W_i) = h_{RD1}(\gamma_i, x_i, y_i, tx_i, ty_i, s_i, o_i)$$
$$= (\gamma_i, x_i, y_i, ty_i, tx_i, s_i, o_i)$$
$$h_{RD2}(W_i) = h_{RD2}(\gamma_i, x_i, y_i, tx_i, ty_i, s_i, o_i)$$
$$= (\gamma_i, x_i, y_i, -ty_i, -tx_i, s_i, o_i)$$

Concerning rotations, we calculate n PIFS for given rotations $\theta = k\tau$ between 0 and $\pi/4$. We note them ifs_k with :

$$\tau = \frac{\pi}{4n}$$

and k an integral number between 0 and n included.

We obtain the other rotations, by applying the reflection properties, thus we can write :

if $\theta = k_1\tau$ with $n \le k_1 \le 2n$, or $\pi/4 \le \theta \le \pi/2$ then,

$$H_{rot(\theta)}(ifs) = H_{RD1}(H_{rot(\theta - n\tau)}(ifs))$$
$$= H_{RD1}(ifs_{k_1 - n})$$

if $\theta = k_1\tau$ with $2n \le k_1 \le 4n$, or $\pi/2 \le \theta \le \pi$ then,

$$H_{rot(\theta)}(ifs) = H_{RV}(H_{rot(\theta - 2n\tau)}(ifs)) = H_{RV}(ifs_{k_1 - 2n})$$

$$= \begin{cases} H_{RV}(ifs_{k_1 - n}) = H_{RV}(H_{RD1}(ifs_{k_{1-2n}})), \\ \qquad \text{si } 2n \le k_1 \le 3n \\ H_{RV}(ifs_{k_1 - n}) = H_{RV}(ifs_{k_{1-3n}}), \\ \qquad \text{si } 3n \le k_1 \le 4n \end{cases}$$

if $\theta = k_1\tau$ with $4n \le k_1 \le 8n$, or $\pi \le \theta \le 2\pi$, then we can use the two last cases :

$$H_{rot(\theta)}(ifs_i) = H_{RH}(H_{rot(\theta - 4n\tau)}(ifs_i)) = H_{RH}(ifs_{k_1 - 4n})$$

k_1 - 4n is between 0 and π.

So PIFS are calculated for some given angles between 0 and $\pi/4$, for other rotations, we just need to apply one or two reflections to an already calculated PIFS.

4.3 Distance calculation

Distance calculation is an important part of the search process. The query time is very dependant on the complexity of the distance.

The distance calculation is based on the PIFS parameters.

The distance calculation between two multiresolution representations is described further.

4.3.1 Distance calculation between two PIFS parameters sets

We consider $ifs_P = \{U_i\}$, i=0,...,n as the set of parameter vectors of the pattern PIFS and $ifs_{Im}\{V_j\}$, j=0,..,m, as the set of parameter vectors of the image PIFS.

Let U_i be a parameter vector of the pattern sample, U_i can be written as:

$$U_i = (\gamma_i\ x_i\ y_i\ tx_i\ ty_i\ s_i\ o_i)$$

We search among the parameter vectors of the image PIFS, a vector Vj, such as U_i and V_j are comparable (same isometrie code and translation coordinates).

For each U_i, we build $\Phi(U_i)$ the set of vector V_j belonging to the image PIFS, which are comparable to U_i.

$$V_j = (\gamma_j\ x_j\ y_j\ tx_j\ ty_j\ s_j\ o_j)$$

V_j verifies:

$$\begin{cases} \gamma_i = \gamma_j \\ tx_i = tx_j \\ ty_i = ty_j \end{cases}$$

The distance d_v between U_i and V_i is:

$$d_v(U_i, V_i) = \sqrt{(s_i - s_j)^2 + (o_i - o_j)^2}$$

The distance d between the pattern PIFS and image PIFS is:

$$d(ifs_E, ifs_{\text{Im}}) = \sum_{U_i \in ifs_E} e(U_i, \Phi(U_i))$$

with $e(U_i, \Phi(U_i)) = 1$ if $\exists V_i \in \Phi(U_i)$ such as $d_v(U_i, V_i) < \varepsilon$

ε is a similarity threshold between U_i and V_j.

To calculate the ratio of vector U_i in the set of vector V_j, we use :

$$p(ifs_E, ifs_{\text{Im}}) = \frac{d(fs_E, ifs_{\text{Im}})}{card(ifs_E)}$$

4.3.2 Distance calculation between two multiresolution representations

Each graph of multiresolution representation can be written as a set of parameter vectors. The distance between two multiresolution representations, is the minimum distance between two set of parameter vectors.

U_i, is a vector of the graph G_n. It represents the relation $R^{(n)}$ between an element $p^{(n)}$ and an element $q^{(n)}$ of the image partition at the level *n*:

$$U_i = (x_i\ y_i\ tx_i\ ty_i\ s_i\ o_i v_i)$$

- x_i, y_i position of $p^{(n)}$ in the image ;
- tx_i, ty_i translation to move $q^{(n)}$ to $p^{(n)}$.
- s_i, o_i and v_i, values attached to $R^{(n)}$.

Distance between two vectors is computed as described previously.

4.3.3 Robustness - Human perception

Two close images, in the PIFS parameters space are not close in the R^2 space (considering the root mean square distance, for instance). The human perception of similarity is different to the similarity in the PIFS parameters space. This is due to the noise sensitivity of the compression process. The alteration of a range block can modify several PIFS parameter vectors.

To increase the robustness of the system a range block is associated to several domain blocks, during the compression process. These domain blocks are sorted (by similarity to the range block) into a list.

The set ifs_P is a set of vector lists $\tilde{W}_i$. Several parameter vectors W_k are associated to a range R_i.

$$\tilde{W}_i = \left(W_O, W_1, \ldots, W_l\right)$$

W_k are chosen such as:

$$rms\left(W_k\left(D_m\right), R_i\right) < \varepsilon_l,\ k = 0,1,\ldots,l$$

rms is the root means square distance. The elements $\tilde{W}_i$ are sorted:

$$rms\left(W_k\left(D_m\right), R_i\right) < rms\left(W_{k+1}\left(D_{m'}\right), R_i\right),\ k = 0,1\ldots,l$$

4.4 Search scheme

The compressed pattern contains several PIFS or multiresolution representations to cover all possibilities of rotation or scale.

The distance calculation between different pattern PIFS gives us a ratio distribution. The ratio or score associated to the image is the maximum of the distribution.

4.4.1 Partition fitting

The image and pattern partitions can be different. To be able to retrieve the pattern, we have to consider all possible partitions of the pattern. However, the adapted compression scheme described in the indexing part, allows to reduce the number of required partitions.

The number of different required partitions is:

$$x_{\text{range}} \cdot y_{\text{range}}$$

if range blocks ($x_{\text{range}} \cdot y_{\text{range}}$ pixels) are square blocks.

For each partition we calculate a set of scores corresponding to the distance between each pattern PIFS and the image PIFS. The score associated to the image, is the maximum of this set.

4.5 Data structure

The distance calculation between the pattern compressed data and the index image involves to scan the vector set $\{U_i\} \times \{V_j\}$. To optimise this process, we reduce the number of V_j vectors and we use a indexed data structure to store the PIFS.

We consider a X×Y pixels pattern. Only the parameter vectors V_j, which satisfy:

$$tx_j \le X \text{ and } ty_j \le Y$$

are used for the distance calculation.

This reduction allows to spare memory space and process time, when the pattern size is smaller than the domain research space used during the image indexing.

The PIFS parameters can be organised as a tree (see *figure* [9]). This representation allows a direct access to a vector V_j by using its isometrie code and translation coordinates.

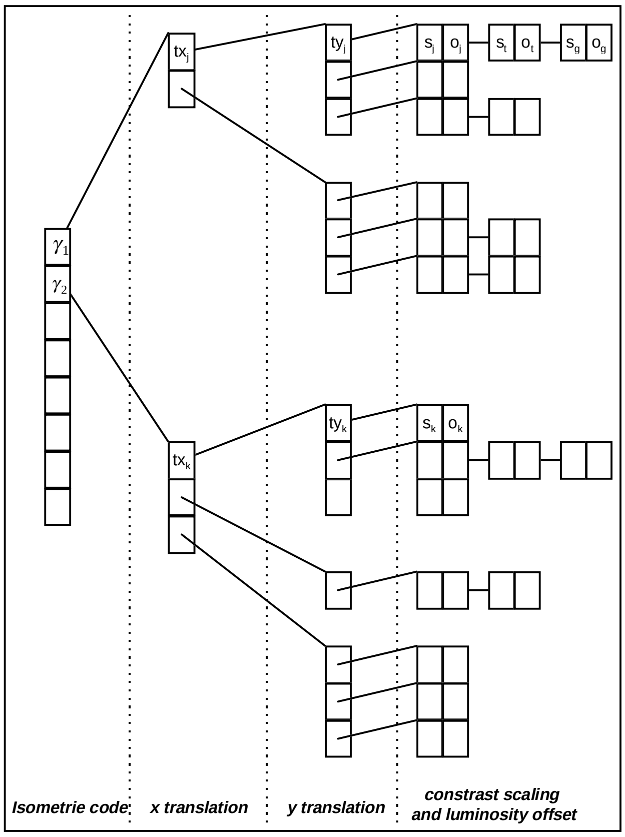


Figure *[9] : Tree representation of the PIFS.*

5. Results

Some results are given in the *tables* [3][4] and figures [10][11]. For instance, it takes 0.75 seconds to find a 32x32 pixels pattern in a 100 images database (using mono-resolution representation of the pattern). When using the multiresolution or multicompression representation of the pattern, queries processing times are longer (see *table* [4]).

Some compression times, using multiresolution or multi-compression are given in *table* [5].

We remark that the query processing times are linear to the pattern size.

Pattern size	Image number	Search time
32 x 32 pixels	100	0.75 s
64 x 64 pixels	100	2.90 s
128 x 128 pixels	100	11.60 s

Table *[3]: Queries computation times, using a mono-resolution of the pattern (computed on a SUN* SPARC 20).

Pattern representation	Search time
Multiresolution (3 levels)	7.52 s
Multi-compression	
3 different sizes	6.34 s
4 different orientations	8.42 s
3 different sizes and 32 orientations	25.20 s

Table *[4]: Queries computation times, using a multiresolution or multi-compression representation of a 64x64 pixels pattern. Search process in a 100 images database (computed on a SUN* SPARCstation 20).

6. Conclusion

In this paper, a method for indexing and content based retrieval based on pattern matching, has been proposed. As compared to other known methods, it is computationally attractive.

This method can be improved by combining it with other approaches (based on colours, shapes or texture) to perform a first selection of candidate images.

Further works will consists in optimising the search process and implement this application (Fractal compression, indexing and search processes) on a parallel computer (SIMD architecture dedicated to image processing developed in our laboratory, *SYMPHONIE*).

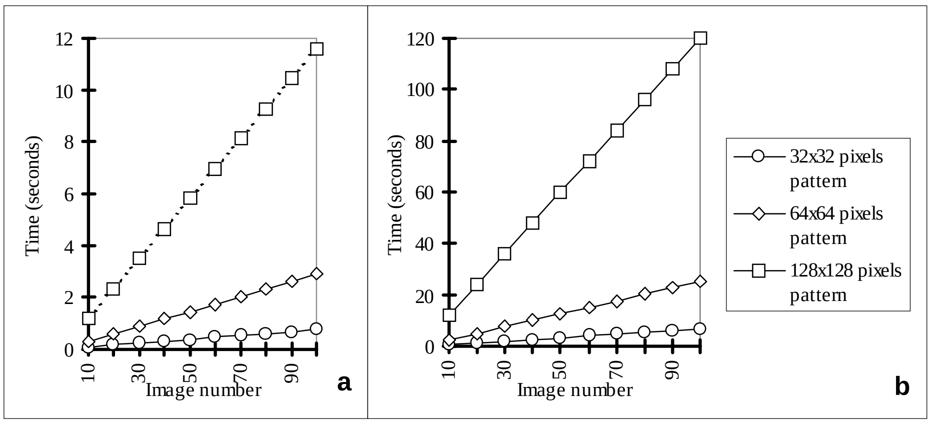
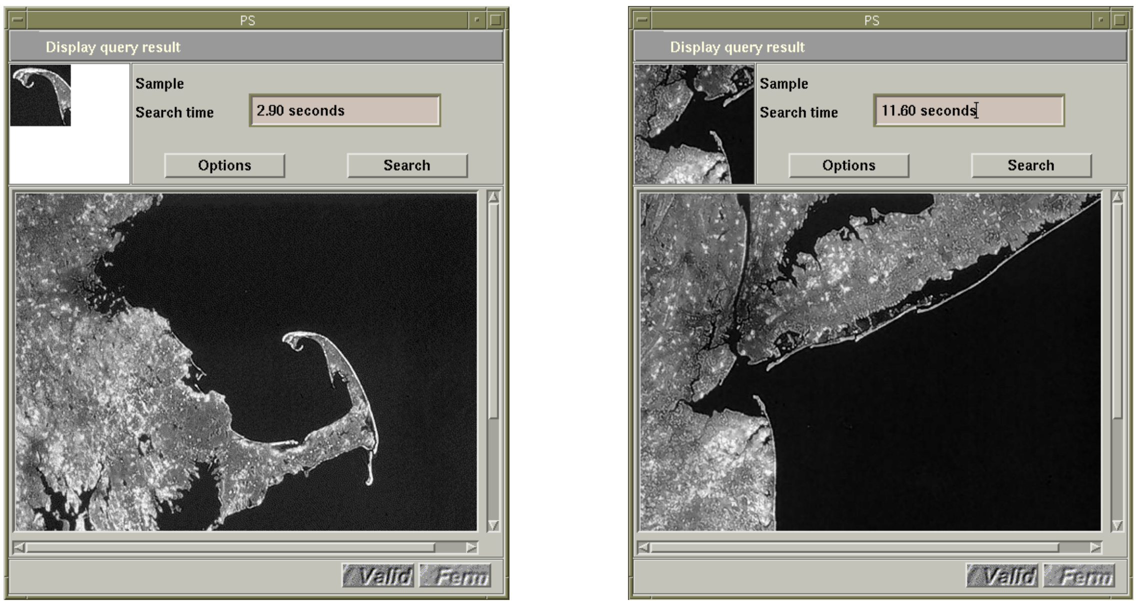


***Figure* [10]: (a) Search time for various patterns (mono-compression representation of the pattern). (b) Search time for various patterns, using a multi-compression representation of the pattern (3 different sizes and 32 different orientations) (computed on a SUN *SPARCstation* 20).**



Sample

Search time 2.90 seconds

Sample

Search time 11.60 seconds

Figure *[11]: Results of a query based on a 64x64 pixels pattern (left) and on a 128x128 pixels pattern (right). The database contains 100 images (computed on a SUN* SPARCstation *20) (Landsat images).*

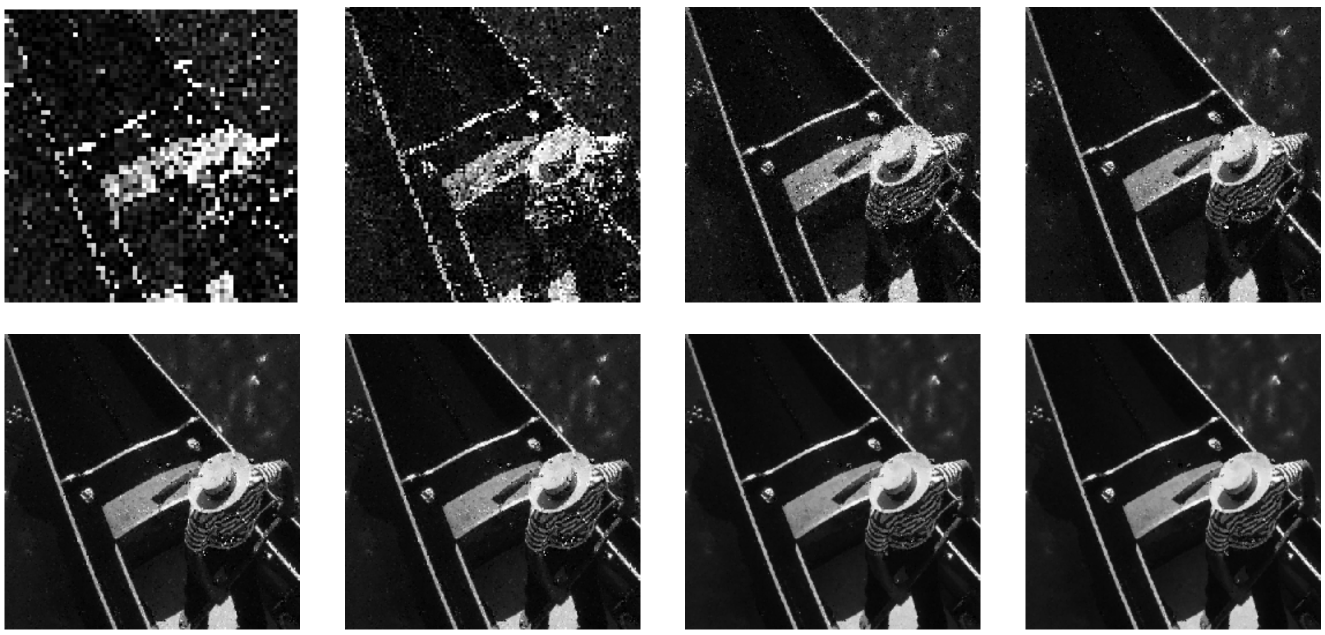


Figure *[12]: Decompression of an image, encoded with a Fractal compression. Successive iterations of a PIFS (Fractal compressed code for the image "*gondole*"), the original image appears.*

7. References

1. Y. Fisher. "Fractal Compression: Theory and Application to Digital Images", *Springer Verlag, New York 1994.*

2. A. Jacquin. "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformation", *IEEE Transaction on Image Processing, 1992, Vol. 1, N° 1.*

3. C. Frigaard, J. Gade, T. T. Hemmingsen, T. Sand. "Image Compression Based on a Fractal Theory".

4. P. Aigrain, H. Zhang, D. Petkovic. "Content-based Representation and Retrieval of Visual Media: A State-of-the-Art Review", *Multimedia Tools and Applications special issue on Representation and Retrieval of Visual Media*

5. A. Pentland, R.W. Picard, S. Sclaroff. "Photobook: Content-Based Manipulation of Image Databases", *International Journal of Computer Vision, Fall 1995*

6. R. Mehrotra, J.E. Gary. "Similar-Shape Retrieval In Shape Data Management", *Computer September 1995*

7. W. Niblack, R. Barber, W. Equitz, M.D. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, G. Taubin. "QBIC Project: querying images by content, using color, texture, and shape", *Storage and Retrieval for Image and Video Databases*

8. B. Scassellati, S. Alexopoulos, M.D. Flickner. "Retrieving images by 2D shape: a comparison of computation methods with human perceptual judgments", *Storage and Retrieval for Image and Video Databases*

9. D. Tegolo. "Shape analysis for image retrieval", *Storage and Retrieval for Image and Video Databases*

10. B.V. Funt and G.D. Finlayson. "Color constant color indexing", *Technical report, School of Computing Science, Simon Fraser University, Vancouver, B.C. Canada 1991.*

11. M.A. Stricker. "Color and geometry as cues for indexing", *Technical report, Department of Computer Science, The University of Chicago, Nov. 1992*

12. F. Arduini, S. Fioravanti, D. Giusto. "Natural Surface Characterization by Multifractals", *MVA'90, IAPR Workshop on Machine Vision Applications, Nov 1990.*

13. N. Sarkar, B.B. Chaudhuri. "An efficient approach to estimate fractal dimension of Textural Images", *Pattern Recognition, Vol 25,* No *9, pp 1035-1041, 1992.*

14. J.M. Marie-Julie - H. Essafi. "Fast parallel multimedia data base access based on wavelet multiresolution pyramidal decomposition", *MVA'96, IAPR Workshop on Machine Vision Applications.*

15. J.M. Marie-Julie, P. Adam, D. Juvin. "Real Time Stereovision Using Correlation on a Parallel SIMD Computer, Sympati 2", ICA3PP, Brisbane Australia 19-21 April, 1995.