Texture-based Video Indexing

Bertrand Guilheneuf, Jacques Lévy Véhel

TEXTURE BASED VIDEO INDEXING

Jacques Lévy Véhel

Projet Fractales, INRIA Rocquencourt
Domaine de Voluceau. B.P. 105
78153 Le Chesnay, France
and Ircyn, BP 92101, 1 rue de la Noé,
44321 Nantes Cedex 3, France
jlv@bora.inria.fr

Bertrand Guiheneuf

Projet Fractales, INRIA Rocquencourt
Domaine de Voluceau. B.P. 105
78153 Le Chesnay,
France
guiheneu@bora.inria.fr
http://www-rocq.inria.fr/fractales

ABSTRACT

We present a method for indexing and searching video sequences based on textural information. Our method proceeds by first defining and computing texture descriptors relevant to the database at hand. Then a similarity distance is computed between a video sample given by the user and small space-time regions in the sequences. The system returns an ordered set of best matches. We show some results on real sequences, which indicate that this scheme has reasonable complexity and performs well in practice. We believe that these good results stem mainly from a) the introduction of a new texture descriptor based on dynamic local Hölder exponents, and b) the automated adapted choice of the relevant parameters.

Keywords : Multimedia, Video Indexing, Texture.

1. INTRODUCTION

Indexing and searching video sequences have become matters of paramount importance with the development of multimedia computing and the World Wide Web. While many methods have been proposed for indexing e.g. faces or simple manufactured objects, much less work has been devoted to highly complex structures such as ones present in textured regions of sequences [1, 7, 8][1]. A first explanation of this fact is that it is simply not as easy to formulate a request that deals with texture as it is for simple objects. Nevertheless, textures and specially moving textures are an essential part of the semantic content of images, and, in some cases, textural criteria are crucial for determining a given sequence. One could for instance be interested in scenes that contains rivers flowing in an rapid and chaotic way, or, on the contrary very quietly. The same type of requests could be made concerning clouds in the sky, or movement in the branches of a tree. A second reason is of course that indexing textures in an efficient way is a difficult problem, and that none of the various approaches give fully satisfactory results. Indeed, it is already a hard task to have a simple and general enough definition of a texture. Depending on the definition chosen, one then uses different geometric and/or statistical characterizations, which only capture partly the complexity of the real data. As a consequence, although most works on image indexing mention texture as a relevant information, very few systems actually use it in real applications.

Our aim in this work is to propose a method for indexing and searching video sequences based on requests that pertain to a dynamical textural information. Our approach displays the following features:

- The indexing step makes use of a large number of texture descriptors (several hundreds) and selects the most pertinent ones *for a given database* through a learning strategy.
- Request are made by examples, i.e. the user selects on a particular sequence a space-time region for which he is interested in finding a similar textural content.
- Temporal information is explicitly taken into account for characterizing a texture.

We have implement our method in an algorithm, called Sumi, which proceeds in two steps. The first is off-line: Texture descriptors are computed on the database, and a relevant subset of these descriptors is stored in an index. This is described in details in the next section. In the second, online, step, when presented with a request from a user, Sumi computes a similarity distance between the given image and the elements of the database, and returns an ordered list of best matches. This part is explained in section 3. Finally, section 4 shows the results of some numerical experiments.

[1] By textured regions, we mean parts of the video sequences where information lies e.g. in water, the sky, trees or more generally natural landscapes, but also complicated manufactured objects such as facades of some buildings, fabrics, etc...

2. INDEXING STEP

To build an index which is both meaningful and of reasonable size for a given database, we proceed in two steps: first, a large number of texture descriptors are computed on each sequence. Then, using data analysis techniques, the most discriminating ones are selected and gathered to constitute the index. For both steps, we used the same strategy as in the texture analysis software called ARTHUR developed at Inria [6]. This software allows to classify and segment images on the basis of textural information, and has been used with success for automated recognition tasks on still images such as satellite images of the earth or MR medical images. We now proceed to describe precisely the indexing step.

Phase 1: Computation of the texture descriptors

Each sequence in the database is first cut into elementary space-time regions of fixed size, typically $32 \times 32 \times 10$, where the two first figures refer to a number of pixels and the last one to a number of frames (the sequences in the database are subsampled in time so that there are 10 frames per second). On each region, the following texture features are computed:

- *Statistical parameters.* These are based on co-occurrence matrices [5], from which 8 classical descriptors are extracted: energy, entropy and contrasts. Recall that the coefficient $a_{i,j}$ of the co-occurrence matrix C_T is the proportion of pairs of pixels (P_1, P_2) such that the grey level at P_1 is i, the grey level at P_2 is j, and the $(2D + t)$ translation between P_1 and P_2 is T. These descriptors are quite classical in texture analysis, and generally give good results for homogeneous textures such as grass, water or wool. We consider roughly 20 translations, which yield approximately 150 co-occurrence parameters. Another class of statistical parameters is based on the use of an autoregressive model for the texture in the $32 \times 32 \times 10$ region. The three parameters of the AR model are used as texture descriptors.

  For both classes, it was found that using the additional information brought by time increased very much the performances. Since a direct implementation in $2D + t$ does however also increase greatly the computational burden, a specific fast algorithm was designed in this case.

- *Time-Space-Frequency parameters.* We compute a Gabor decomposition of each region, i.e. a set of coefficients whose square describes the amount of energy in the image around a given location in time, space and frequency [3]. In addition, the time-space-frequency atoms used for the analysis have a preferred orientation, which allows to characterize anisotropic textures. Such parameters are useful to discriminate between textures which display varying structures at different resolution, or which are not homogeneous. Examples include wood, fabrics with repetitive patterns or facades of buildings. We use 8 different frequencies bands and 8 orientations on each region. For each of the 64 filtered images thus derived from the original region, we compute the energy and entropy. This results in a total of 128 parameters.

- *Fractal parameters.* Textures are by definition very irregular regions. It thus seems relevant to try to classify them according to some regularity indices. These indices must be local, so that they can capture the variation in space and time of irregularity. In that view, we have used local Hölder exponents α[4]. For a still image, these are roughly defined for each 32×32 region by comparing the oscillation in the region (i.e. the total variation of the grey levels) with functions of the form $\epsilon \rightarrow \epsilon^{\alpha}$. For a sequence, we compute a dynamic exponent $\alpha(t)$ by centering a 3D window of size 32x32x6 around each frame. Such descriptors are specially relevant for discriminating e.g. water or clouds with slow or rapid movements, and scale invariant textures. We provide in the Appendix more details about the precise definition and computation of local Hölder exponents.

- *Color parameters.* Color is an important aid to discriminate between textures otherwise indistinguishable. An example is leaves texture on a tree at different periods of the year, e.g. spring and autumn. We use simply the mean, energy and entropy in each band R, G and B (HSV coding would also do), which yields a total of 9 parameters. The time dimension was not found to be important for color features, so that, contrarily to all parameters described above, the color characterization does not take into account the time varying nature of the searched textures.

The total number of parameters computed in this phase is of the order of 300. For a 10 seconds sequence of images at resolution 352×288 subsampled at 10 frames per second, the computational time is of the order of a few minutes on a PC. The total size for storing all the parameters is a little less than one third of the size of the original sequence.

Most of the time, many of the parameters are not very discriminating, and the information they provide is largely redundant. In order to build an efficient index, i.e. an index which is compact and contains parameters which are both discriminating and "independent", we use data analysis techniques described below.

Phase 2: Selection of the most discriminating descriptors

What is classically done in texture recognition is the following: one computes a series of parameters on a learning

set, which contains several instances of each texture. The discriminating power of each parameter is then evaluated according to a procedure that takes into account the number of cases where a texture has been misclassified. In addition, the correlation to other parameters is measured. This allows to extract a set of robust and "independent" parameters that serve to classify new images.

Unfortunately, such an approach cannot be used in our case, since we do not know *a priori* which textures will be present in the database. Indeed, it would be a severe restriction to allow the user to search only for textures belonging to a predefined limited set. We have thus adopted a different strategy. What we need to do is obtain an index reasonable in size by discarding parameters which are not relevant. Two main factors render a descriptor useless: it may have poor discriminating power in the considered database and/or be very much correlated to another descriptor which has better discriminating capability. This leads naturally to the following criteria:

- A parameter is considered good if it has high variability on the database. This ensures that the parameter holds at least some information allowing to discriminate regions.
- A parameter is considered good if it has low correlation with the other ones.

The selection then proceeds by first ranking the descriptors by decreasing order of variance to obtain a list $D_1, \ldots, D_n$. Then, the following operations are performed sequentially for $j = 2, \ldots, n$: a) compute the correlation of D_j with all its predecessors, b) if the max of these correlations exceeds a given ratio r, remove D_j from the list. The result is the index sought for. r is set such that the total size of the index is of the order of one tenth of the size of the database. Note that this strategy allows to adapt the size of the index to the complexity of the database. Also, it yields an automated adaptive scheme which allows a blind selection of the best parameters for any given database.

In most of the tests that were performed, it was found that the fractal parameters based on the behaviour of the dynamic local Hölder exponents were selected in priority, i.e. they were the ones with best discriminating power. This was in particular the case for sequences containing water, clouds, tree branches or grass, moving with different velocity. Time-space-frequency parameters usually came second, and were found particularly efficient for building facades. To our knowledge, this is the first time that (dynamic) local Hölder exponents are used for indexing. Our results thus show that this new technique is indeed relevant, as it improves on more classical ones in most cases.

3. SEARCHING STEP

The searching proceeds as follows. The user first chooses a space-time block in the database. More precisely, he selects a 32×32 region in a particular image and the system understands that the user is interested in the dynamical textural content of this region around this particular frame. Usually the time neighbourhood is of size 10, i.e. 5 frames are considered on each side of the selected one. This allows to discriminate between e.g. slowly and rapidly moving water. The parameters of the chosen space-time region are then retrieved, and compared to the parameters of all other regions in the database: Let $D^* = (D_1^*, \ldots, D_p^*)$ be the vector of texture descriptors for the region selected by the user, and $D^k = (D_1^k, \ldots, D_p^k)$ be the vector corresponding to the $k - th$ region in the database. For each k, we compute a distance $\mathcal{D}_k = d(D^*, D^k)$, where d is a given distance function. The program then returns the list of regions ordered by increasing magnitude of $\mathcal{D}_k$. We tested several distance functions. Best results seem to be obtained when one uses a simple weighted L^2 norm and the Kullback distance (weights are necessary because all the parameters do not have the same importance).

4. EXPERIMENTAL RESULTS

We tested our method on a database composed of 40 video sequences of outdoor scenes. Strong textural information contained in these sequences include flowing water, moving clouds, facades of buildings, trees, grass, stones, ... Each sequence lasts approximately thirty seconds, the images being 352×288. In step 1/phase 1, all the parameters are computed. This took a little less that an hour. Phase 2 selects approximately 100 parameters, which constitute the index. Thus the additional amount of storage for the index is roughly one tenth of the size of the original database. Step 2 (the searching) is fast: each query is processed in around 0.1 second (all times are given for a computation on a PC).

We display in figure 1 an example of a query and the associated result. The top image is the one that serves to formulate the query by example. The image is split into regions of size 32×32, and the user selects a zone whose texture he is interested in, here a particular facade of a building. The image at the bottom shows the best match found in *another* sequence. This example show that Sumi is able to retrieve quite finely detailed textures.

As a second example, we show in figures 2 and 3 another reference image with selected texture, along with the four best matches found by Sumi. Again, we display only one image per matching sequence in order to have a meaningful illustration. Since the delicate color texture of the sky might not appear clearly on the printed paper, the interested reader might look at an electronic version on our site: http :// www-rocq.inria.fr/fractales/.

These results show that Sumi indeed allows for robust and efficient video indexing based on textural information.

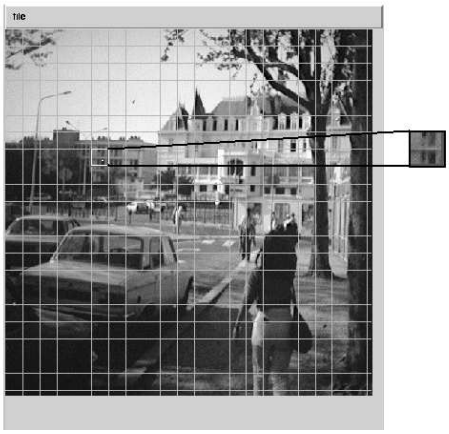

Figure 1: Image on which the user has selected a region for search of similar textural information in the database (top). Image in another sequence in which there is a region whose textural content yields the best match to the request (bottom).

Figure 2: Image containing the query.

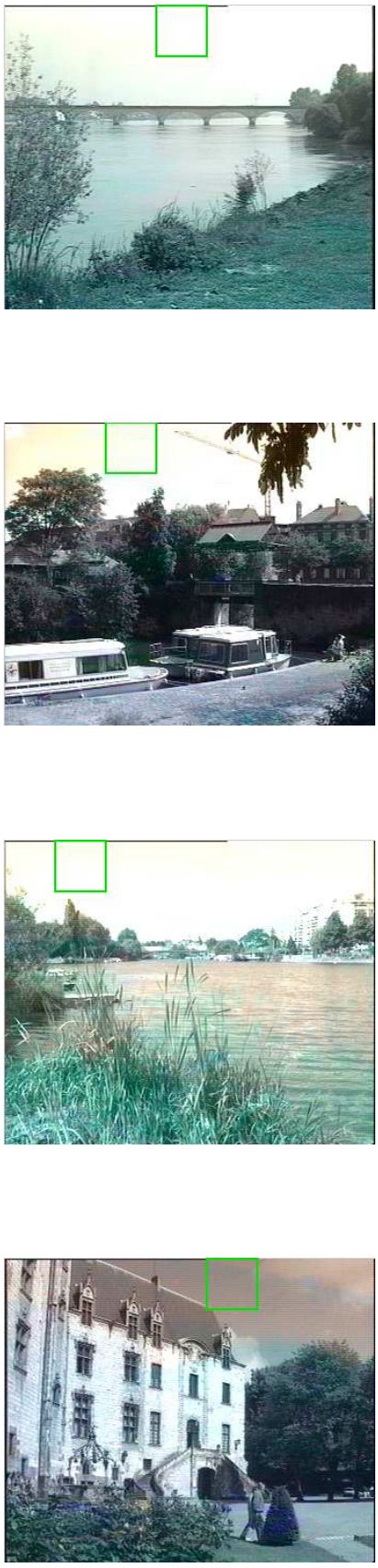

Figure 3: Four best matches in different sequences.

Acknowledgements

This work was supported by the french CNET under grant #96ME28

5. APPENDIX : LOCAL HÖLDER EXPONENTS

We describe in this section some basic facts about local Hölder exponents. The interested reader is refered to [4] for more details. For the sake of notationnal simplicity, we shall give the definitions in 1D. Extension to 3D is straightforward.

Hölder exponents are tools akin to evaluating the regularity of a function $f : K \to \mathbb{R}$ where K is a bounded subset of $\mathbb{R}$. There are basically two ways to define a meaningful Hölder exponent :

Definition 1 *Pointwise Hölder exponent*

Let α be a positive real number, $\alpha \notin \mathbb{N}$, and $x_0 \in \mathbb{R}$. A function $f : \mathbb{R} \to \mathbb{R}$ is in $C^{\alpha}_{x_0}$ if there exists a polynomial P of degree less than α such that:

$$|f(x) - P(x - x_0)| \leq c|x - x_0|^{\alpha}. \quad (1)$$

When $\alpha \in]0, 1[$, this reduces to:

$$|f(x) - f(x_0)| \leq c|x - x_0|^{\alpha} \quad (2)$$

The pointwise Hölder exponent of f at x_0, denoted $\alpha(x_0)$, is the supremum of the α for which (1) holds.

The pointwise Hölder function of f is defined as:

$$\alpha(x) = \sup \{\alpha : f \in C^{\alpha}_{x}\}.$$

This regularity characterization is widely used in fractal analysis because it has direct interpretations both mathematically and in applications. It has been shown for instance [2] that α_f indeed corresponds to the auditive perception of smoothness for voice signals. Similarly, simply computing the Hölder exponent at each point of an image already gives a good idea of its structure, as for instance its edges or the roughness of its textures at any particular point. More generally, the pointwise Hölder exponent is useful in all applications where it is desirable to model, synthetize or process signals which are highly irregular, and for which the relevant information lies in the singularities more than in the amplitude : this is in particular exactly the case for the problem of indexing textures in video sequences. However, the pointwise Hölder characterization has also a number of drawbacks. A first one is that it is not stable under the action of (pseudo) differential operators. This precludes, for instance, the use of the Hilbert transform, commonly used in signal processing. In the same way, knowing the pointwise Hölder exponent of a function at a point x_0 is not sufficient to predict the Hölder exponent of its derivative at the same point. Second, this exponent is generally hard to compute numerically, in particular when the data at hand do not have a sufficiently high resolution, as is typically the case for video sequences.

An alternative solution is to consider the local Hölder exponent: Let $\alpha \in]0, 1[$, $\Omega \subset \mathbb{R}$. One says that $f \in C^{\alpha}_{l}(\Omega)$ if:

$$\exists C : \forall x, y \in \Omega : \frac{|f(x) - f(y)|}{|x - y|^{\alpha}} \leq C$$

Then, let:

$$\alpha_l (f, x_0, \rho) = \sup \{\alpha : f \in C^{\alpha}_{l} (B (x_0, \rho))\}$$

$\alpha_l (f, x_0, \rho)$ is non increasing as a function of ρ.
We are now in position to give the definition of the local Hölder exponent :

Definition 2 *Let f be a continuous function. The* local Hölder exponent *of f at x_0 is the real number:*

$$\alpha_l (f, x_0) = \lim_{\rho \to 0} \alpha_l (f, x_0, \rho)$$

This exponent is stable under differentiation or integration. Moreover, and this is crucial for our application, it is easier to estimate than the pointwise Hölder exponent. In particular, it is possible to obtain reasonable estimates even for video sequences [2]. Finally, in most cases, the local Hölder exponent corresponds just as much to the intuitive perception of regularity as the pointwise one does, so it is in general equally useful for our purpose of discriminating time-varying textures.

6. REFERENCES

[1] J.F. Cullen, J.J. Hull, P.E. Hart, "Document Image Database Retrieval and Browsing Using Texture Analysis", *ICDAR97*, 1997

[2] K. Daoudi, J. Lévy Véhel, Y. Meyer, "Construction of continuous functions with prescribed local regularity", J. of Constructive Approximation,**014**(03), pp. 349-385, 1998.

[3] H. G. Feichtinger, T. Strohmer (Ed.), Gabor Analysis and Algorithms, Birkhauser, 1998

[4] B. Guiheneuf, J. Lévy Véhel "Multifractal Image Denoising", *SCIA*, 1997.

[5] R. M. Haralick, Statistical and Structural Approaches to texture, Proc. IEEE, **67**(5), 786-804, 1979.

[2] There is a profound mathematical reason why the local Hölder exponent is easier to estimate. This is however a bit technical, and we refer the reader to [4] for a precise statement.

[6] P. Mignot, J. Lévy Véhel “ARTHUR : un Système d’Analyse de Textures” *TS* **9**(6), pp. 507-516, 1993.

[7] T. Randen, J. Husoy, “Image Content Search by Color and Texture Properties”, *ICIP97*, 1997

[8] E. Saber, A.M. Tekalp, “Integration of Color, Shape, and Texture for Image Annotation and Retrieval”, *ICIP96*, 1996