Scalable Feature Extraction for Coarse-to-Fine JPEG 2000 Image Classification

Antonin Descampe*, *IEEE Member,* Christophe De Vleeschouwer, *IEEE Member,* Pierre Vandergheynst, *IEEE Member,* and Benoit Macq, *IEEE Fellow,*

Abstract

In this paper, we address the issues of analyzing and classifying JPEG 2000 code-streams. An original representation, called *integral volume*, is first proposed to compute local image features progressively from the compressed code-stream, on any spatial image area, regardless of the code-block borders. Then, a JPEG 2000 classifier is presented, that uses integral volumes to learn an ensemble of randomized trees. Several classification tasks are performed on various JPEG 2000 image databases and results are close or even better than the ones obtained in the literature with non-compressed version of these databases. Finally, a cascade of such classifiers is considered, in order to specifically address the image retrieval issue, i.e. bi-class problems characterized by a highly skewed distribution and by a large amount of test samples compared to learn samples. An efficient way to learn and optimize such cascade is proposed. We show that staying in a JPEG 2000 framework, initially seen as a constraint to avoid heavy decoding operations, is actually an advantage as it can benefit from the multi-resolution and multi-layer paradigms inherently present in this compression standard. In particular, unlike other existing cascaded retrieval systems, the features used along our cascade are increasingly discriminant and lead therefore to a better complexity *vs* performance trade-off.

EDICS Category: ARS-SRE

A. Descampe, C. De Vleeschouwer, and B. Macq are with the Communications Laboratory (TELE), Louvain School of Engineering (EPL), UCL, Place du Levant, 2, 1348 Louvain-la-Neuve, Belgium (e-mail: {antonin.descampe, christophe.devleeschouwer, benoit.macq}@uclouvain.be).

P. Vandergheynst is with the LTS2 laboratory, Electrical Engineering Institute, EPFL, ELB 111 (Building ELB) Station 11, CH - 1015 Lausanne Switzerland (e-mail: pierre.vandergheynst@epfl.ch).

I. Introduction

Today's imaging applications have to deal with an ever-growing amount of digital data [1]. Management and exploitation of this information requires (1) an efficient and flexible representation of the images, and (2) application-specific techniques exploiting this flexibility to guide and assist the user in his search in the data space. Such techniques could be useful in any application that involves (semi-)automatic image categorization or retrieval tasks, such as audiovisual archives retrieval, computer-assisted medical diagnosis, remote sensing on satellite images, etc.

In this paper, we focus on a compressed scalable image representation, namely JPEG 2000, and investigate how discriminant features can be extracted from such embedded code-stream to perform image categorization and retrieval tasks. Being able to search for relevant images directly within compressed bit-streams (or through a partial decompression) has an obvious advantage in terms of computational load. Moreover, we show in this paper that JPEG 2000 is actually well suited for image retrieval tasks. Indeed, its various scalability levels allow for true and efficient coarse-to-fine searches among the bit-streams. Furthermore, the JPEG 2000 wavelet transformation has also a strong discriminant power and is suited to analyze image characteristics like edges and textures.

Among the broad range of imaging applications involving classification or retrieval tasks, we will focus on those where areas of interest are characterized by an ensemble of different textures. Texture has to be understood here in the broad sense of a "set of local neighbourhood properties of the gray levels of an image region" [2]. "Grays levels" is even too restrictive in our case as color information (at least in the low frequencies) will also be exploited in the retrieval process. While this choice might appear to exceedingly narrow the set of potential applications, we will show that many objects can actually be characterized by an ensemble of textures. This is even truer if the relative spatial positions of the textures composing an object are taken into account, as it will be the case in the proposed system. Focusing on texture-based retrieval techniques, we will not consider shape matching algorithms, like the ones presented (in a JPEG 2000 framework) in [3]. However, these algorithms could efficiently complement the proposed retrieval system and are, as such, an interesting future research direction, as explained in the conclusion of this paper.

II. Overview

Figure 1 presents an overview of the proposed system. Its goal is to determine if an image, stored in the JPEG 2000 format, is similar to a set of images of interest. Practically, for each image that has to be analyzed, discriminant features are extracted from its JPEG 2000 code-stream. Several

successive features sets are extracted, each one with an extraction cost slightly higher than the previous one, but also with a potentially higher discriminant power. A first classifier uses the first features set, and feeds a second classifier with the processed images that are likely to be positive. The second classifier uses the two first features sets and passes the image to the third classifier if needed and so on. If an image is very different from the image(s) of interest, it will be identified as such very early in the procedure and will be quickly discarded. On the contrary, a relevant image will pass the successive classifiers and reach the end of the cascade. By doing so, the amount of data that has to be decoded from the JPEG 2000 code-stream is directly related to the relevance of the image. We show in this paper that JPEG 2000 is particularly well suited for this kind of coarse-to-fine classification.

It should be noted that in the following experiments, the classifiers have been trained based on a learning set (ensemble of labeled images) in an offline process. This is a first step towards a user-centered approach with relevance feedback and query-by-example method: in such framework, the user would progressively build the learning set by giving positive and negative examples and labeling examples selected by the system, while the system would progressively build the classifiers based on the user feedback.

A. Contributions

Three main contributions are presented in this paper. We now introduce each of these ideas briefly and describe them in details in subsequent sections.

The first contribution is a **method to extract, process and store a hierarchy of features from a JPEG 2000 code-stream**. Both header-based [4] and wavelet-based [3] features are considered. To store all the extracted JPEG 2000 features values, we extend the concept of integral image first introduced in [5] and more recently made famous by Viola and Jones in [6]. We propose to represent a JPEG 2000 code-stream with a chain of *integral volumes*. This representation draws benefit from the JPEG 2000 scalability levels and allows for a very fast features computation on any image area.

The second contribution is the combination of integral volumes mentioned above with an ensemble of random decision trees [7] to build an **efficient and robust JPEG 2000 image classifier**. Based on the work by Marée *et al.* [8], we take advantage of the integral volume structure to generate random samples in the images. We show that chosen features lead to satisfying

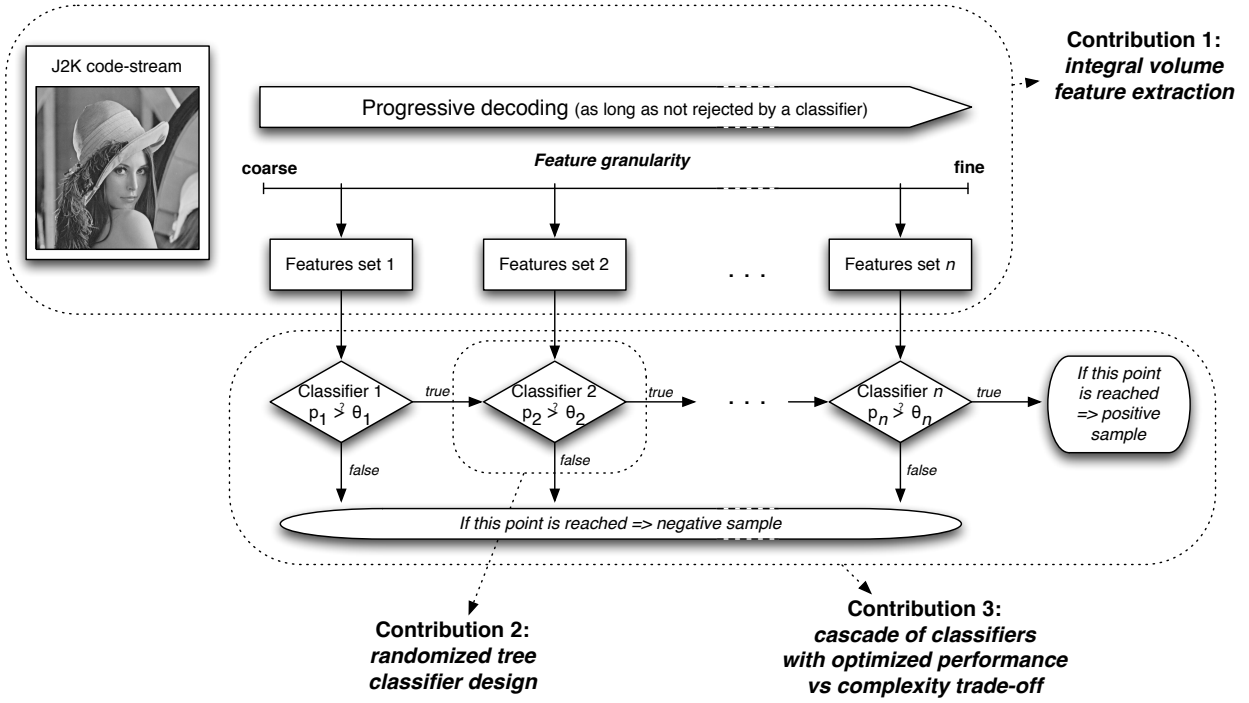


Fig. 1. System overview. Starting from a JPEG 2000 image, discriminant features are progressively extracted from it and feed a cascade of classifiers. As long as the image is not considered as negative by the cascade, each classifier i will compute a probability p_i that the image is positive and compare it to a threshold θ_i $(i = 1, .., N)$.

classification results on several different image databases.

The last contribution is the tight integration of the JPEG 2000 scalability levels for designing a **cascade of JPEG 2000 image classifiers**. Cascade of classifiers are well suited for image retrieval tasks and object detection. When one has to dig into a compressed image database, the challenge is to retrieve relevant areas while minimizing the amount of decompressed data. In this work, starting from the coarse classification that can easily be obtained from header-based features, we use the resolution and bit-depth scalability of JPEG 2000 to progressively extract more information and refine the classification as we go deeper into the cascade. For a given image area, the amount of data that will need to be entropy-decoded is therefore directly related to the relevance of this area in the retrieval process. This idea is directly inspired from the work of Geman and Fleuret [9] and from the one of Viola and Jones [6]. Concurrently to [10], we proposed in [11] to improve the Viola and Jones cascade by taking into account the extraction cost of a feature. This leads us to an

interesting "complexity[1] *vs* performances" trade-off when designing a cascade of classifiers.

B. Paper organization

The remainder of the paper is organized as follows. Section III first gives some JPEG 2000 key elements and then details which features are used and how they are extracted from JPEG 2000 code-streams. In particular, the scalability of the extraction process is underlined. We also introduce the concept of integral volume to store these features. In Section IV, integral volumes are first combined with random decision trees to set up a JPEG 2000 classifier. Then, in Section V, several of these classifiers are cascaded to address image retrieval problems in a computationally efficient way. Section VI presents our experimental results. We first show the discriminant power of our JPEG 2000 classifier. Then, a texture retrieval problem is introduced to illustrate the effectiveness of the cascade approach. Finally, Section VII summarizes the paper and gives future research directions.

III. Scalable feature extraction

Numerous papers have been written on the use of JPEG 2000 code-streams for classification or retrieval tasks. Each of them proposes a set of features that can be more or less easily extracted from the code-stream and that is used in a classification process, most often based on some kind of distance metric. As we shall see, the proposed approach uses features with similar discriminant power than the ones proposed in the related literature. However, their extraction can take advantage of the JPEG 2000 scalability to trade-off between the classification accuracy and the amount of data that has to be extracted from the code-stream. Moreover, in order to address more realistic classification problems, we propose an efficient way to process and store these features, called *integral volumes*.

A. JPEG 2000 key elements

We now review the elements of the JPEG 2000 compression standard that are relevant to understand the rest of the paper. Interested readers may refer to [12], [13] for more details.

In short, JPEG 2000 is based on successive dyadic discrete wavelet decompositions of the image components, whose subbands are partitioned into *code-blocks* that are coded independently. Each

[1]This complexity being computed as the total amount of data that has to be decompressed in the retrieval process.

code-block is entropy-coded into an embedded bitstream, i.e. into a stream that provides a representation that is (close-to-)optimal in the rate-distortion sense when truncated to any desired length. This implies in particular that wavelet coefficients are coded bit-plane by bit-plane, from the Most Significant Bit (MSB) to the Least Significant Bit (LSB).

In final, any JPEG 2000 code-stream is organized as a succession of packets, each packet containing data referring to a certain resolution, a certain image component, a certain spatial zone, and a certain range of bits inside the coefficients bit-depth (this last element is referred to as a SNR scalability). It is therefore possible to decode a JPEG 2000 code-stream in many different ways, and without having to process all the packets. To obtain a low resolution version of the image for example, one has simply to extract the related low resolution packets.

B. Feature types

Two kind of features can basically be extracted from a JPEG 2000 code-stream: the ones from the packet headers that do not involve any partial decompression, and the ones based on the wavelet coefficients that require an entropy-decoding step.

1) Header-based features: as mentioned above, a JPEG 2000 code-stream is made of a succession of packets. Each packet has a header that contains relevant information about its content. In particular, for a code-block c present in a packet, we can extract B_c, the number of bytes used to entropically encode the image data contained in code-block c, and M_c, the maximum number of significant bit-planes[1] in code-block c. The former reflects the efficiency with which the entropy coder did compress the wavelet coefficients from this code-block. As such, it can be seen as a measure of the source entropy and gives therefore a good indication on the informational content of the code-block [4]. The latter corresponds to the $\log_2$ of the maximum modulus of the wavelet coefficients present in the code-block. In [14], Mallat showed that this maximum modulus describes well the kind of singularity present in the image. It has been used for example in [15]. In terms of extraction complexity, these features are of great interest as they do not imply any (partial) decompression of the JPEG 2000 code-stream. In comparison with a decompression operation, the computational complexity of the header processing step is negligible. In the following, we will

[1]A significant bit-plane is a bit-plane which either contains at least one non-zero bit or is below another significant bit-plane

consider only the number of entropy-coded bytes B_c, as experiments have shown little improvement when using both kind of features.

2) Wavelet-based features: research on wavelet-based retrieval techniques has been ongoing for many years now. The spatial-frequency representation provided by the wavelet transform gives important information on the kind of local singularities present in the image [16]. In particular, textures end edges are well discriminated based on the statistical properties of the wavelet coefficients distribution.

[17] proposes to model this distribution with a Generalized Gaussian Density (GGD) and hence, to characterize an image with the vector aggregating the 2 parameters of a GGD in each subband. [18] follows the same approach using Gaussian Mixture Models instead of GGD. Like in [17] and [18], we propose to characterize the probability distribution of the wavelet coefficients by a set of parameters. Starting from a JPEG 2000 code-stream, the extraction process of these parameters obviously implies an entropy-decoding step, as we have to get back to the wavelet coefficients. Unlike [17] and [18], a very simple distribution approximation is used: it does not require any computational overhead in addition to the decoding step and, more importantly, it can be *progressively refined* during the decoding step. Let's consider a code-block c with M_c significant bit-planes. As a bit-plane from this code-block is being decoded, new significant coefficients are counted. For a given bit-plane k ($k = 1, ..., M_c$), the number $S_{k,c}$ of new significant coefficients corresponds to the amount of wavelet coefficients x in the code-block such that

$$2^{k-1} \leq |x| < 2^k.$$

These $S_{k,c}$ ($k = 1, ..., M^c$) values might be considered as the values of an histogram whose non-overlapping bins have variable width. As such, $S_{k,c}$ correspond to an approximation of the wavelet coefficients distribution for the related code-block. Note that this approximation is progressively refined, as bit-planes are being decoded : a new decoded bit-plane will simply split the last bin in two new non-overlapping bins, further refining the distribution approximation. In the following, these bin values will be considered as independent features, each one giving some relevant information on the image.

The header and wavelet-based features described above will be used to feed the proposed image classifier. As the application may require to compare (portions of) images that do not necessarily have the same amount of pixels, the values B_c and $S_{k,c}$ are normalized by the total number N of coefficients present in the sample, to define b_c and $s_{k,c}$, respectively.

C. Scalable extraction

The feature extraction process can take advantage of the scalability inherently present in JPEG 2000, namely the spatial, resolution and SNR scalability levels. It is indeed possible to make a trade-off between the desired characterization accuracy and the feature extraction cost it implies. Let's consider a given spatial area in an image. The spatial scalability of JPEG 2000 allows to select the packets related to this area without having to process the remaining code-stream. Then, a coarse characterization of the area content can be obtained by extracting the header-based features, from the lowest to the highest resolution. If a more accurate description of the image content is required, we can complement the header information with wavelet-based features, at the cost of an higher computation load (as entropy-decoding operations are required). The wavelet coefficient distribution is approximated, from the lowest to the highest resolution. Moreover, for each resolution, this approximation is progressively refined as more bit-planes are decoded (using the SNR scalability).

To illustrate the benefit that can be drawn from such scalable extraction, a texture classification process is presented. Let's consider 4 different classes of textures. Each class contains 40 items, each one being a 640×480 grayscale image. Samples from each class are presented in Fig. 2(a). All images are JPEG 2000-compressed with 4 resolution levels, resolution 1 corresponding to the lowest resolution (the LL-subband). There are three successive steps in this example of classification process, each using a bit more information than the previous one.

In the first step of the classification process, only very cheap features are used, namely header-based features. As we see on Fig 2(b), this information is already sufficient to discriminate the "water" class from the three other ones. For the second step, in addition to the header information, we now assume that the 2 most significant bit-planes (7th and 8th) of resolution 1 and 2 have been decoded. As shown in Fig. 2(c), this new information allows to clearly separate the "glass2" samples from the two remaining classes, although it was not possible to do so with header-based features only. In the last step, the third most significant bit-plane (the 6th) from all resolutions is supposed to be decoded. Fig 2(d) shows how the two last classes can be separated.

As we see, this scalable extraction perfectly fits with a classification process made of several cascaded classifiers. This will be detailed in Section V and used in an image retrieval context.

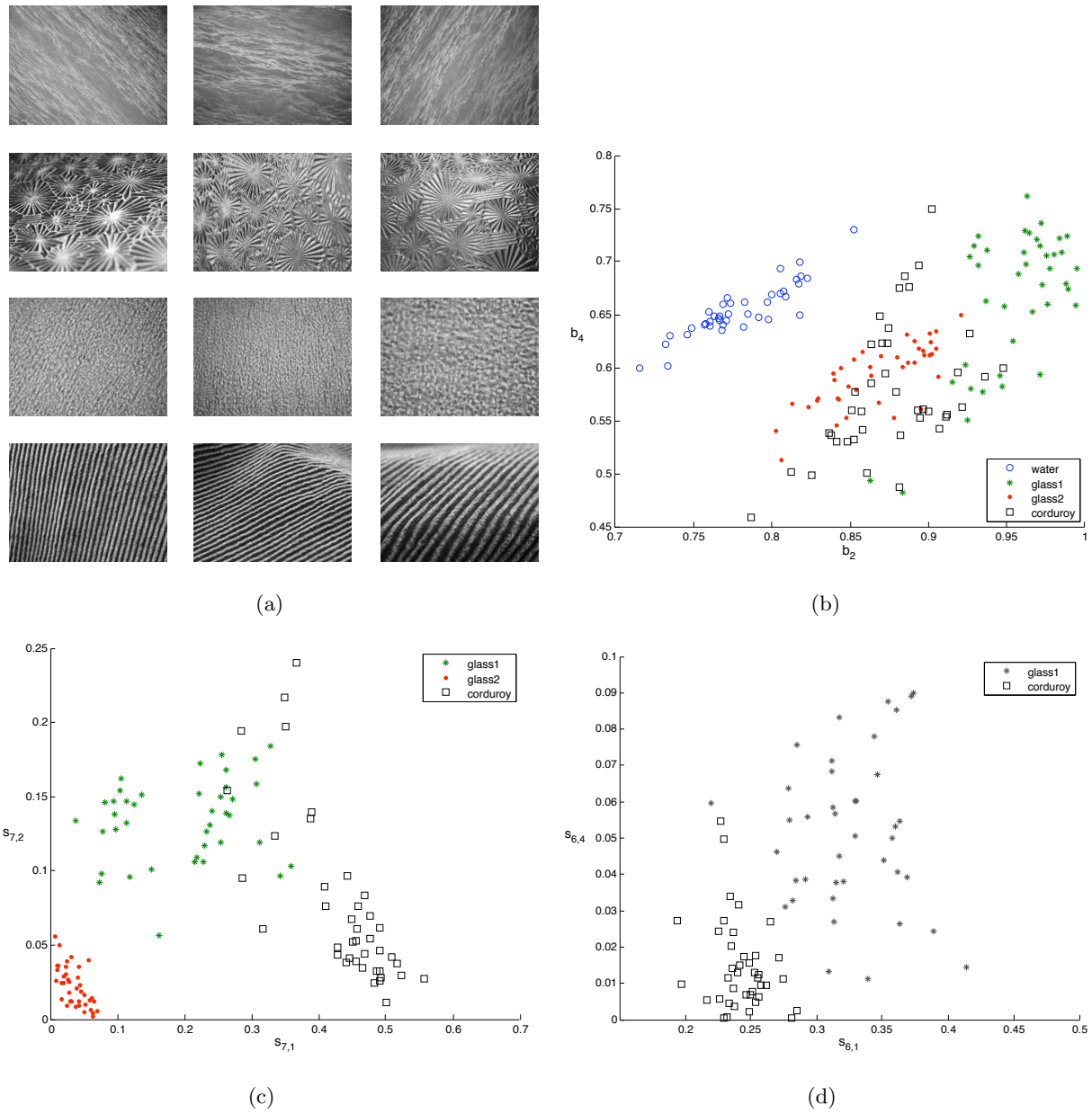


Fig. 2. A scalable classification process. (a) Texture samples randomly selected from 4 different classes, one in each row: water, glass1, glass2 and corduroy. Source : Ponce texture database [19]. (b) The first step uses only header information and is able to discriminate the "water" class from the three other ones. (b_i = normalized number of entropy-coded bytes for resolution i). (c) In the second step, the two most significant bit-planes from resolution 1 and 2 have been decoded and allow to separate textures from class "glass2". ($s_{j,i}$ = fraction of new sign. coeff. in bp j, for res. i). (d) The third and final step allows to more clearly discriminate the two remaining classes, "glass1" and "corduroy".

D. Integral volume representation

Let's summarize what we do have up to this point. Considering a code-block c of a JPEG 2000 compressed image, the relevant information that can be extracted about it is (i) b_c, the normalized number of bytes used to entropically encode code-block c, and (ii) $s_{k,c}$, the fraction of code-block coefficients that become significant in the bit-plane k ($k = 1, .., M_c$). Each of these extracted values gives a global information on the code-block. If the area of interest is made of several code-blocks, these values are very simply aggregated.

A drawback of the features described above is that we have to stick to code-blocks borders when selecting an area of interest. All code-blocks in a JPEG 2000 code-stream have indeed the same size, which is typically around 32x32 or 64x64 pixels. At low resolution levels, such amount of pixels can cover a very large spatial area, which can reveal itself to be very constraining if we want to extract relevant information from a smaller spatial zone.

To solve this problem and extract features very rapidly, on any spatial area, independently from code-blocks borders, we propose an intermediate representation of the image, called *integral volume*. This representation is a direct extension of the *integral image* introduced in [5] and more recently made famous in [6]. Starting from an image P where $P(i, j)$ is the pixel value at coordinates (i, j), the integral image I_P is defined as

$$I_P(i, j) = \sum_{i' \leq i, j' \leq j} P(i', j') \tag{1}$$

Having computed this integral image I_P allows to very quickly compute the sum of all pixel values in any rectangular area contained in P [6]. Let (i_r, j_r) denote the coordinates of the top-left corner of a given rectangle r in P, and w and h the width and height of r. The sum of pixels $S_{P,r}$ in rectangle r is simply given by

$$S_{P,r} = I_P(i_r, j_r) + I_P(i_r + w, j_r + h) - I_P(i_r + w, j_r) - I_P(i_r, j_r + h) \tag{2}$$

In this paper, we propose to use, for each bit-plane k from each code-block c, the integral image concept described above, where the matrix of pixel values P has been replaced by the significance map $\sigma_{k,c}(i, j)$:

$$\begin{aligned} \sigma_{k,c}(i, j) &= 1 \quad \text{if } c(i, j) \geq 2^{k-1} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

where $c(i, j)$ refers to the coefficient value at coordinates (i, j) in code-block c. By doing so, we obtain a table $T^s_{k,c}$ such that each element $T^s_{k,c}(i, j)$ at column i and row j is the number of

significant coefficients in code-block c at bit-plane k, among all coefficients above and to the left of element (i, j), inclusive:

$$T^s_{k,c}(i,j) = \sum_{i' \leq i, j' \leq j} \sigma_{k,c}(i', j') \text{ , for } i = 1, ..., W_c \text{ and } j = 1, ..., H_c$$

where W_c and H_c are respectively the width and height of code-block c.

Interestingly, computation of table $T^s_{k,c}$ perfectly fits with the entropy-decoding procedure of bit-plane k. Indeed, the significance status of each coefficient of a code-block is a state variable of the JPEG 2000 entropy decoder. As such, it is constantly maintained and updated during the decoding procedure. Hence, once bit-plane k has been decoded, $\sigma_{k,c}(i, j)$ is directly available and $T^s_{k,c}(i, j)$ is obtained by the following recursive formula

$$T^s_{k,c}(i,j) = T^s_{k,c}(i-1,j) + T^s_{k,c}(i,j-1) - T^s_{k,c}(i-1,j-1) + \sigma_{k,c}(i,j).$$

Initial conditions are set based on the previously decoded adjacent code-blocks, as shown in Fig. 3. Hence, we obtain an integral table $T^s_{k,sb}$ related to the whole subband sb.

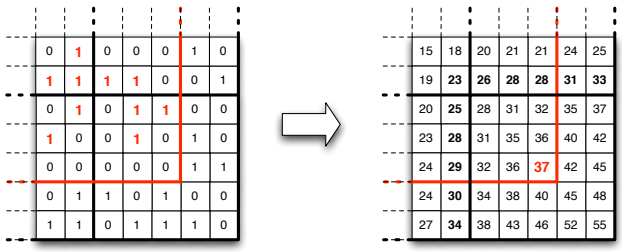


Fig. 3. Integral table $T^{sc}_{k,c}$ with adjacent code-blocks taken into account. Initial conditions are set to the border lines values of the code-block currently decoded (black bold numbers in the figure).

By stacking the tables $T^s_{k,sb}$ for all bit-planes of subband sb, we obtain what we call an *integral volume* of this subband. Each element of such volume is simply given by

$$V_{sb}(i,j,k) = T^s_{k,sb}(i,j) \quad for \quad \begin{array}{l} i = 1, ..., W_{sb} \\ j = 1, ..., H_{sb} \\ k = 1, ..., M_{sb} \end{array} \tag{3}$$

where W_{sb} and H_{sb} are respectively the width and height of subband sb. M_{sb} is the maximum number of bit-planes in the subband. Once computed, this integral volume allows to very easily

obtain the normalized number $s_{A,K}$ of new significant coefficients in bit-plane range K ($K = [k_0; k_1]$, $k_0 \leq k_1$), on any spatial area A, defined by its upper left corner (i_0, j_0) and its lower right corner (i_1, j_1), inside the subband sb:

$$\begin{aligned} s_{A,K} &= (V_{sb}(i_0, j_0, k_0) + V_{sb}(i_1, j_1, k_0) \\ &\quad -V_{sb}(i_0, j_1, k_0) - V_{sb}(i_1, j_0, k_0) \\ &\quad -V_{sb}(i_0, j_0, k_1) - V_{sb}(i_1, j_1, k_1) \\ &\quad +V_{sb}(i_0, j_1, k_1) + V_{sb}(i_1, j_0, k_1)) / (i_1 - i_0) . (j_1 - j_0) . \end{aligned} \tag{4}$$

In order to have a single representation from which it would be possible to extract both wavelet and header-based features, we use the same concept of integral table to store the b_c values (defined on page 10). Assuming that any coefficient belonging to code-block c is encoded using $\beta_c = \frac{B_c}{W_c.H_c}$ bytes per coefficient, we define the integral table T_c^b as

$$T_c^b(i, j) = T_c^b(i-1, j) + T_c^b(i, j-1) - T_c^b(i-1, j-1) + \beta_c(i, j)$$

for $i = 1, ..., W_c$ and $j = 1, ..., H_c$. Like $T_{k,c}^s$, table T_c^b can be extended to T_{sb}^b related to the whole subband. By adding table T_{sb}^b on top of all tables $T_{k,sb}^s$ of the integral volume V_{sb} defined in Eq. 3, the normalized number of bytes used to encode all coefficients in area A is given by:

$$\begin{aligned} b_A &= (V_{sb}(i_0, j_0, M_{sb}+1) + V_{sb}(i_1, j_1, M_{sb}+1) \\ &\quad -V_{sb}(i_0, j_1, M_{sb}+1) - V_{sb}(i_1, j_0, M_{sb}+1)) / (i_1 - i_0) . (j_1 - j_0) . \end{aligned} \tag{5}$$

Eventually, to increase the robustness to rotation changes, integral volumes from the 3 different subbands of a same resolution level are merged by an averaging operation:

$$V_r(i, j, k) = \frac{\sum_{sb} V_{sb}(i, j, k)}{3}$$

In conclusion, starting from any JPEG 2000 image, compressed with N_L decomposition levels, an intermediate representation called *integral volume* can be progressively built, as more data is decompressed. It consists in several different volumes of size W_rxH_rx$(M_r + 1)$ $(r = 0, .., N_L)$. With this data structure available, one can easily get the normalized number of bytes $b_{A,r}$ used to encode a spatial area A in resolution r, and the normalized number of new significant coefficients $s_{A,K,r}$ for a range of bit-planes K inside resolution r of spatial area A .

IV. A JPEG 2000 image classifier

In our work, we propose to combine an ensemble of random decision trees with our JPEG 2000 integral volume representation. Tree-based methods consist in a succession of answers to (binary) questions that progressively divide (and classify) an initial set of items [20]. Randomization [21], when applied to decision trees, can drastically reduce the variance and overfitting problems generally affecting decision trees [7]. Practically, we assume the existence of two sets of images, each image belonging to one of N_c distinct classes ($N_c > 2$)[1]. The first set is called the learning set, and is used to train the classifier. The second set, called the test set, is used to evaluate the performances achieved by the trained classifier. Our design of classifier involves three successive steps that we detail hereunder.

A. Sampling

In this paper, we define a sample as the basic element that will be classified by the ensemble of random decision trees. An entire image could be considered as a single sample but it appears that subsampling images has several advantages when dealing with classification tasks. The main one is that it increases the robustness of the classifier: all samples from an image do not need to be correctly classified to still make a good decision about the class of the image. Like in [8], the subsampling operation consists in selecting square subwindows of random sizes and at random positions, each subwindow inheriting its class from the parent image. Once a sample has been defined, a certain amount of features can be extracted from it, characterizing the sample content. These features will be used in the tree nodes to route the sample towards a certain tree leaf.

Thanks to the integral volume representation described above, the subsampling operation only consists in randomly generating pairs of coordinates, each pair corresponding to the upper left and lower right corners of a sample. It should be noted that this samples generation is very cheap and can thus easily be renewed from one tree to the next one, which is similar to a bagging process [22]. Then, once a sample has been defined inside the image, feature values $b_{A,r}$ or $s_{A,K,r}$ are extracted from it by specifying a resolution r, a bit-plane range K and a spatial zone A inside the sample, as described at the end of Section III, on page 12. Unlike [8], there is no "fixed-size feature vector" characterizing each sample. Actually, there is virtually an unlimited number of features that can be extracted from a given sample.

[1]The case ($N_c = 2$) is assimilated to an image retrieval task and will be envisioned in Section V.

B. Learning

During the learning phase, samples are used to train an ensemble of extremely randomized trees [7] (Extra-Trees). In an ensemble of T Extra-Trees, candidate splits are chosen completely at random. This means that both the tested feature and the cut point selected for thresholding, are chosen randomly. The split eventually chosen is either the best one among K candidates, or the first one to reach a minimum score Sc_{min}. The score measure is based on the entropy reduction brought by the split. The splitting process is repeated recursively until a node contains less than n_{min} samples. A leaf l from a tree t is then characterized by a probability distribution $p_t(i|l)$ $(i = 1, ..., n_c)$, the probability that a sample reaching leaf l belongs to the ith class among n_c classes. This distribution is simply obtained by counting, for each class, the training samples reaching this leaf.

C. Testing

During the test step, N_{test} samples are extracted from each test image and are propagated in the T trees. Each sample reaches a specific leaf in each tree. Once all N_{test} samples from the test image k have been propagated in the T trees, we obtain a set $L_k = \{l_k\}$ of reached leaves on image k. The probability $p(i|L_k)$ that image k belongs to class i is estimated by

$$p(i|L_k) = \frac{1}{N_{test}} \sum_{l_k \in L_k} \frac{\sum_{t=1}^{T} p_t(i|l_k)}{T}$$

and the class c_k^* predicted for test image k is simply

$$c_k^* = \arg \max_{c_k \in [1;n_c]} p(c_k|L_k).$$

V. Coarse-to-fine image classification: a cascade of image classifiers

In this Section, we consider a typical image retrieval problem, for which there are only two classes (positive and negative images) that are highly asymmetric (there are much more negative images than positive ones). We exploit the JPEG 2000 scalability, to limit as much as possible the amount of information that has to be partially decompressed when searching a large dataset.

To take into account the class asymmetry, a natural idea that comes up is to first try to reject the most obvious negative images rather than trying to find directly the few positive events. In this way, we are left with our positive images disseminated in a much smaller amount of negative ones. Of course, this subset of images is harder to discriminate but as it is much smaller, we can afford to spend more computational load on each of its images. This simple idea has lead to the *cascade*

of classifiers concept [6], [9], [23]. As shown in Fig. 1 on page 4, a cascade of classifiers is made of a succession of stages, each of them being fed by the images identified as positive by the previous stage. For a given image, each stage i will compute a probability p_i that the image is positive and compare it to a threshold θ_i. Formally, an image retrieval system has two objectives: maximize the detection rate and minimize the false positive rate. Such cascade framework allows to relax the latter one. Indeed, if a negative image is accepted, next stages still have the opportunity to reject it. The succession of stages will then guarantee a small global false positive rate, by multiplying the succession of moderate intermediate false positive rate.

The concept of cascading classifiers is not new and has already been investigated in the literature. Among others, Viola and Jones proposed a cascade of boosted classifiers based on simple rectangle features for robust real-time object detection [6]. Each stage is a weighted sum of single-feature-thresholding classifiers, trained through an AdaBoost process. In a more recent work [24], Brubaker *et al.* questioned the relevance of single-feature-thresholding classifier and showed that the overall performance could benefit from stronger weak classifiers like CART (Classification And Regression Trees). In our work, we push this idea one step further by using an ensemble of random decision trees for each stage. Such random decision tree is a stronger weak classifier than thresholding on a single feature, but is also much easier and faster to build than a classic CART. Moreover, in our implementation, we did not consider any boosting algorithm, which further reduces the complexity of the learning phase.

Since most images are stored in a compressed format, investigating *compressed* image retrieval techniques makes a lot of sense. The proposed system follows the general cascade concept introduced above and integrates it in a JPEG 2000 framework, as explained in Section IV. Using the resolution and SNR scalability levels present in JPEG 2000, successive stages of our cascade are based on an increasing amount of data extracted from the code-streams. The first stages exploit header information at almost no cost to achieve a coarse classification of all images. Having discarded the most obvious negative images, the next stages can afford a slightly more expensive (but also more accurate) classification process on the remaining positive candidates, by decoding their most significant bit-planes. Finally, the last stages will focus on the very best positive candidates and will try to reject those negative images that are the most similar to positive ones. To do so, more (or even all) bit-planes of these positive candidates are decoded so as to obtain the finest approximation of their wavelet coefficients. As we see, the proposed system uses a true **coarse-to-fine approach**: for a given image, the amount of data that will need to be extracted

(and entropy-decoded) is directly related to the relevance of this image in the retrieval process.

An important point to note about the proposed JPEG 2000 cascade is the increasing discriminant power of the features sets used in successive stages. Indeed, most other cascade systems use the same set of features to generate the succession of classifiers. In this latter case, the higher accuracy of classifiers comes only from a harder training set. In our proposal, we not only provide harder images to a classifier but also better tools to discriminate them.

Of course, using richer features implies an higher extraction cost. Hence, simultaneously to the work in [10], we have proposed in [11] to incorporate the feature extraction cost in the global cascade optimization. Indeed, the best cascade will be obtained through a global trade-off between the number of stages, the thresholds used, and the kind and number of features involved. In [24] and [25], the optimization of this trade-off is tightly coupled with the training phase. This allows to get very close from a global optimal solution as training and optimization are mutually updating each other. However, it comes at a price of a high computational cost as training operations are included in optimizing loops and must be repeated many times. In [26] and in the system proposed in this work, the best thresholds are found in a separate process, after the cascade design. As the final adopted thresholds are not necessarily the ones used for training, it obviously implies a suboptimality, but not sufficiently important to justify the huge increase in computational load caused by iterations.

Practically, our image retrieval system is obtained in two successive steps: building and optimization, as described below. Once the cascade has been built and optimized, it can eventually be used to scan a test set.

A. Building

During the building phase, a cascade of classifiers is generated, based on a given learning set. In our approach, we fix *a priori* the number N of stages, together with the part of the JPEG 2000 code-streams that will be available in each stage. This is done based on the relative computational cost of the features.

For each stage, an ensemble of random decision trees is learned. This process is the same as the one described in Section IV, with the noticeable restriction that the number of classes is always 2. The learning set used to generate each stage is thus made of positive and negative images. The set of n_+ positive images is the same in each stage while the n_- negative images change from one stage to the other. Moreover, an image from the negative set dedicated to train stage i is used in

this stage only if it successfully passed stages 1 to $i-1$. In this way, successive stages are trained with increasingly difficult learning sets.

B. Optimizing

Once the cascade has been built, the optimization phase aims at finding a threshold $\theta^*_{i,opt}$ for each stage i of the cascade such that the set $\mathbf{T}^*_{opt} = \{\theta^*_{i,opt}\}$ $(i = 1, .., N)$ leads to the best global complexity-performance trade-off. To do so, we use an *optimization set*, different from the *learning set*. In particular, positive images used for optimization have to be different from the ones used for learning. Otherwise, as tree nodes have been selected to best isolate these specific samples, the optimization process could be biased and select too high thresholds. For a given set of thresholds $\mathbf{T}_{opt}$, the performances of a cascade are evaluated according to its classification error and to its cost in terms of computational load.

The classification error is measured by two values, namely the *precision* and the *recall*. Compared to the more common *detection rate* and *false positive rate* leading to the classical ROC curve, precision and recall are defined in the following way:

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP+FN}, \\ \text{Precision} &= \frac{TP}{TP+FP}, \\ \text{Detection rate} &= \frac{TP}{TP+FN}, \\ \text{False positive rate} &= \frac{FP}{TN+FP}. \end{aligned}$$

where TP, FP, TN and FN are respectively the number of True Positive, False Positive, True Negative and False Negative images. Precision and recall are preferred to the classical ROC curve because these measures are more suited for highly skewed datasets, as explained in [27]. For a given set of thresholds $\mathbf{T}_{opt}$ leading to Precision P and Recall R, the global classification error E is then given by

$$E(\mathbf{T}_{opt}) = 1 - [\lambda_{prec}.P(\mathbf{T}_{opt}) + (1 - \lambda_{prec}).R(\mathbf{T}_{opt})] \tag{6}$$

where λ_{prec} is fixed *a priori* and represents the relative importance of the precision compared to the recall ($0 \leq \lambda_{prec} \leq 1$). $E(\mathbf{T}_{opt})$ is a value comprised between 0 and 1, a better cascade leading to a value closer to 0.

The computational cost measures the fraction of bits that need to be entropically decoded to make a decision on all images. A cost $C(\mathbf{T}_{opt}) = 1$ means that all bit-planes from all

resolutions levels of all images have to be decoded. For the sake of simplicity, the cost of header information extraction is considered equal to the cost of decoding one bit-plane. Practically, the cost computation is done in the following way: for each stage i, a cost factor c_i is computed. This coefficient represents the portion of an image that will have to be decoded if it reaches stage i. It takes into account the number of resolution levels (and their respective size), subbands, components and bit-planes that are involved in stage i. As these numbers are fixed *a priori* when designing a cascade, the c_i coefficients are constant values for a given cascade. For example, a stage i that uses header information and the 2 most significant bit-planes out of 8, from all resolution levels, subbands and components will be given a cost factor $c_i = \frac{3}{9} = 0.333$. The global cost is then given by

$$C(\mathbf{T}_{opt}) = \sum_{i \in [1,N], T_{opt,k} > 0} n_i.(c_i - c_{i'}) \tag{7}$$

where n_i is the number of samples that enter stage i ($i = 1, ..., K$) and i' refers to the closest previous stage with a threshold > 0 ($c_{i'} = 0$ if there is no such stage). The dependency on the set of thresholds $\mathbf{T}_{opt}$ comes obviously from the fact that a threshold change will affect the n_i's. It should be noted that Eq. 7 assumes that each stage will at least re-use the data already decoded in previous stages. Moreover, for a given stage i, if $\theta_{i,opt}$ is taken equal to 0, this stage is considered to be skipped by the classification process and no cost is counted for this step. The number of stages defined in the building step is therefore only a maximum value as the optimization step could realize that better performances can be obtained by skipping some of these stages.

Based on Eq. 6 and 7, the global cascade inefficiency for a given set of thresholds $\mathbf{T}_{opt}$ is measured by the following expression

$$\begin{aligned} I(\mathbf{T}_{opt}) &= (1 - \lambda_{cost}).E(\mathbf{T}_{opt}) + \lambda_{cost}.C(\mathbf{T}_{opt}) \\ &= (1 - \lambda_{cost}).\left(1 - (\lambda_{prec}.P(\mathbf{T}_{opt}) + (1 - \lambda_{prec}).R(\mathbf{T}_{opt}))\right) \\ &\quad + \lambda_{cost}.C(\mathbf{T}_{opt}) \end{aligned} \tag{8}$$

where λ_{cost} represents the relative importance of the computational cost compared to the classification error ($0 \leq \lambda_{cost} \leq 1$). Consequently, the cascade optimization process consists in finding a set of thresholds $\mathbf{T}^*_{opt}$ such that

$$\mathbf{T}^*_{opt} = \underset{\mathbf{T}_{opt} \in \mathbb{R}^N_{[0,1]}}{\arg\min}\ I(\mathbf{T}_{opt}) \tag{9}$$

Eq. 8 shows that the optimization process is influenced by two parameters: λ_{prec} and λ_{cost}, that balance the relative importance of cost, precision and recall. This formulation could be used in a

Lagrange-multiplier approach to solve a constrained optimization problem such as "Maximize the detection rate under the constraints of a limited amount of false positive samples and a bounded cost" [28].

Based on given λ_{prec} and λ_{cost} values, the optimization problem in Eq. 9 can be solved with an iterative algorithm. Each iteration is made of N steps: during step i, the algorithm seeks the threshold for stage i that will give the lowest inefficiency I, while all other thresholds are fixed to their "up-to-this-point best" value. The number of iteration is limited by an upper bound *maxiter* but can be smaller if the global performance gain between two successive iterations is smaller than a given Δ_{stop}. The initial set of thresholds is taken equal to the set used for learning.

C. Scanning

Once the cascade has been designed (*building* step) and the best set of thresholds $\mathbf{T}^*_{opt}$ has been found (*optimization* step), the image retrieval system can be tested on a test set to assess its real performance. In our experiments, we fed the cascade with randomly extracted sub-images from a pool of negative and positive test images.

VI. Experimental validation

In this Section, several experiments are presented in order to validate and assess the image classifier and image retriever introduced in previous sections. Four databases have been used in these experiments: PONCE, ZUBUD, ETH-80 and COIL-100. Except for the PONCE dataset, the databases and experimental protocols used are the same as in [8]. Datasets and protocols description is summarized in Appendix on page 28 for completeness.

A. JPEG 2000 parameters

All images from the databases are first encoded with the JPEG 2000 algorithm using the OpenJPEG library[1]. No "fancy" option values were used during compression as we wanted our system to be able too work with a basic JPEG 2000 codec. Main parameters are:

- *4 resolution levels*. This means 3 successive decompositions. For the COIL-100 database however, we specified only 2 decompositions because the images were already quite small (128x128).

[1]OpenJPEG is an open-source JPEG 2000 library developed and maintained in the Communications Laboratory at UCL: http://www.openjpeg.org.

- *5-3 DWT filter*. This is the lossless filter bank provided in JPEG 2000 part 1. No significant difference could be highlighted concerning the achieved classification performances when using the 9-7 filter bank.
- *Code-blocks of size* 32x32.
- *No quality layers*. No target bit-rates (either intermediate or final) were specified for the compression. This means that all data (i.e., all bit-planes) present in the original images were included in the compressed code-streams. This choice has been made in order to assess the influence of every bit-plane in a classification task. As a result of this study, we show that the last bit-planes could actually be dropped as they have a smaller discriminant power.

Once the images have been compressed, integral volumes are extracted from the compressed code-streams and stored separately. We consider a maximum number of bit-planes equal to 8.

B. Single classifier experiments

Using the classification scheme described in Section IV, we propose to analyze the discriminant power of the JPEG 2000 features introduced in Section III.

Following parameters for the ensemble of random decision trees are used, as they appear to give good results on all datasets (see Section IV-B page 14 for the meaning of these parameters): $T = 10$, $K = 30$, $n_{min} = 2$ (which means that the trees are fully grown), $Sc_{min} = 0.25$.

During the learning phase, a total of 100 000 samples are extracted from the learning images while during the test phase, 100 samples are computed in each test image to decide to which class this image belongs. We invite the reader to refer to [7] for a deeper analysis of the influence of each of these parameters.

To assess the accuracy that can be achieved with the proposed JPEG 2000 classifier, we compare our performances to other results. In particular, we focus on the system proposed by Marée *et al.* in [8]. Our system mainly differs from [8] in the fact that features are extracted from a compressed code-stream and not from the pixel-domain.

There are 5 different experiments: 4 databases including 2 different protocols for the COIL-100 dataset. For each of these 5 experiments, we compare 3 systems:

- *Raw-RSw-ExTr*: this is the classification scheme described in [8]. Random subwindows ("RSw") are extracted from the uncompressed images ("Raw" images), rescaled to a size of 16x16, and used to build an ensemble of Extremely Randomized Trees ("Ex.Tr."). In case of color images, subwindows are also converted from RGB to another color space. In [8], authors chose the

HSV color space as it proved to be more robust to illumination changes than RGB. In addition to HSV, we also consider the YCbCr color space in our work, since it is the one considered during a JPEG 2000 compression.

- *J2K-IVol-ExTr*: this is the proposed classification scheme. It starts from the JPEG 2000 code-streams ("J2K") from which it extracts Integral Volumes ("IVol") before building Extra-Trees.
- *Others*: this refers to the range of other results found in the literature that follow the same evaluation protocols. In each case, the paper corresponding to the best performance is indicated. Note that there is no such other result for the PONCE database.

Table I shows the error rates achieved by each of these systems, according to the protocols described in Appendix.

TABLE I

CLASSIFICATION ERROR RATES ACHIEVED BY THE SYSTEM DESCRIBED IN [8] (RAW-RSW-EXTR) AND BY OUR SYSTEM (J2K-IVOL-EXTR). THE RANGE OF OTHER RESULTS FOUND IN THE LITERATURE IS INDICATED IN THE LAST COLUMN. THE CITATION REFERS TO THE BEST OTHER RESULT.

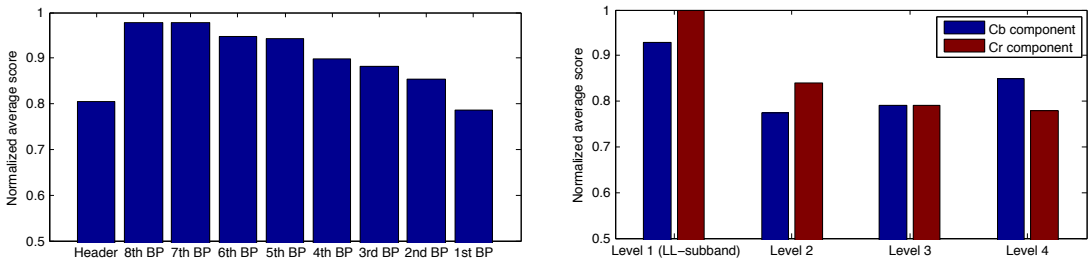
	Raw-RSw-ExTr		J2K	Others
	HSV	YCbCr	IVol-ExTr	
PONCE	45.20%		11.20%	-
ZUBUD	5.22%	4.35%	4.35%	59%-0% [29]
ETH-80	29.09%	27.90%	19.02%	35.2%-13.6% [30]
COIL-100 (1)	14.55%	15.01%	22.51%	50.1%-24% [29]
COIL-100 (2)	0.0033%	0.0024%	0.0017%	12.5%-0.001% [29]

Several interesting aspects of these results can be pointed out.

- The performances achieved by our system are always close from the best result found in the literature. This shows that a JPEG 2000 framework, beside its compression purpose, is well suited to perform classification operations.
- Except for the first protocol of the COIL-100 dataset, the results obtained with the proposed classifier are always equal or better than in [8]. This tends to prove the higher discriminant power of wavelet histogram bins compared to averaged pixel values. In particular, the error-rates obtained for the PONCE dataset underline the much higher ability of the proposed JPEG 2000 classifier to discriminate textures.

- The result obtained for the first protocol of the COIL-100 dataset illustrates the lack of robustness of the proposed classifier in presence of viewpoint changes, compared to [8]. Wavelet histogram bins are more sensitive to a viewpoint change than average pixel values.
- Among the results for the "*Raw-RSw-ExTr*" system, no significant trend could be observed that would urge us to choose between HSV or YCbCr transformation.

All features used in our system do not contribute to the same extent to the image classification task. To evaluate their respective discriminant power, we build an ensemble of totally randomized trees for each dataset. This means that each split is picked totally at random ($K = 1$). As the feature and threshold chosen in each node is totally random, we assume that the average score reached by the nodes using a certain kind of feature is representative of the discriminant power of this kind of feature.



(a) (b)

Fig. 4. (a) Comparison of normalized average scores for the header information and successive bit-planes. (b) Comparison of normalized average scores reached by color components in successive resolution levels. High frequencies (HL, LH, HH) subbands are less discriminant than low frequencies (LL subband).

Three global trends have been observed, whatever the dataset:

- header information is less discriminant than features based on wavelet coefficients histogram bins (Fig. 4(a)),
- more significant bit-planes give more discriminant features (Fig. 4(a)),
- high frequencies from color components are less discriminant than the corresponding low frequencies (Fig. 4(b)).

It should be noted that the decreasing discriminant power of bit-planes is not in contradiction with the fact that adding bit-planes improve the classification performance. The different kinds of feature are complementary, as they discriminate different aspects of the image. This is shown in

Section III-C where the classification accuracy increases as more resolution levels and bit-planes are considered.

As a conclusion, experiments results show that features extracted from JPEG 2000 code-streams are well suited for classification purposes. Initially, we gave us the constraint to stay in a JPEG 2000 framework in order to avoid heavy transcoding operations, speed up image handling, and exploit the algorithm scalability. As the results obtained for several different datasets are in the same range or even better than other systems found in the literature, this constraint actually appears to be also an advantage in terms of classification performance.

C. Cascade of classifiers experiments

In this Section, we assess the performances of the cascade of classifiers described in Section V, which is supposed to exploit the inherent JPEG 2000 scalability. The database used is the PONCE texture database. The main objective is to design a cascade of classifiers able to retrieve samples belonging to a given class c^* among all samples belonging to other classes. Based on the 25 different classes from the original database, we therefore define two image categories: $c_+ = \{images \in c^*\}$ and $c_- = \{images \notin c^*\}$. Different choices of positive class c^* have been tested but we only present here results obtained for class $c^* = 25$ ("Fabric") because achieved performances are representative of the average performance reached with other classes. We arbitrarily fixed the number of stages in the cascade to $N = 3$, each having access to an increasing amount of data compared to the previous one.

1) The first stage uses only header information, from all resolution levels.
2) In addition to header information, the second stage has access to the 3 most significant bit-planes of each resolution level,
3) Eventually, the last stage computes features based on all information: header information and complete decoded wavelet coefficients.

As explained above, each stage is an ensemble of random decision trees. Following settings were used: $T = 50$, $K = 30$, $n_{min} = 2$ (which means that the trees are fully grown), $Sc_{min} = 0.25$. During optimization, we used following values for the parameters: $\Delta_{stop} = 0$, $maxiter = 100$, $\Delta_T = 0.02$.

Concerning the images sets, the database has first been divided in 3 parts: on the 40 images of each of the 25 classes, the first 13 were put in the learning set, the following 13 in the optimization set and the last 14 in the test set.

For the learning phase, 4000 positive sub-images and 10 000 negative ones are used to train each stage. The 4000 positive sub-images are extracted from the 13 positive images and the same set is used to train all stages. Concerning the 13x24 = 312 negative images, they are divided into K ($K = 3$) non-equal sets, the ith set being approximately 2 times bigger than the $(i-1)$th one. From the first set, 10 000 sub-images are extracted to train the first stage. From the second set, sub-images are extracted until 10 000 have passed the first stage and can be used to train the second one. And the same process is made with the third set except that sub-images have to pass the two first stages.

For the optimization and test phases, 4000 positive sub-images and 10 000 negatives ones are simply extracted from the corresponding images and used to optimize and then validate the cascade.

Once the cascade has been built and the thresholds optimized for a given pair of $(\lambda_{prec}, \lambda_{cost})$ (or, equivalently, for a given set of constraints on cost and precision), the test set is used to analyze the performances. Figure 5 compares various Precision-Recall curves. Each curve corresponds to a certain upper bound on the global cascade cost, as indicated in the legend. Points belonging to the same given curve are operating points, each fulfilling the cost constraint and corresponding to a certain trade-off between precision and recall (i.e. a certain λ_{prec} value). This trade-off will be set according to the considered application, depending on the relative importance of false-positive error compared to false-negative error.

The last curve indicated in the legend of Figure 5 is the one corresponding to a single JPEG 2000 classifier (no cascade). In this case, the classifier is trained on the whole learning set and with all features directly available. As it can directly use features from any bit-plane, the global cost is *de facto* equal to 1. Optimization therefore only consists in finding the best trade-off between precision and recall, for a given λ_{prec}. Each such trade-off corresponds to an operating point on the curve.

Although not mentioned on the graph, it should be noted that curves corresponding to cost-targets greater than 0.6 superimpose with the "$cost \leq 0.6$" one. Non-convexity observed at some places on the curves is due to the fact that thresholds for these operating points have been computed on the optimization set while results are given here for the test set.

Based on the curves from Fig. 5, several observations can be made about the performances obtained with our cascade, compared to a conventional single classifier approach.

1) Cost benefit: first of all, from a cost point of view, the benefit of the cascade is obvious: the curve targeting a cost ≤ 0.6 is already very close from the one obtained without any cascade and

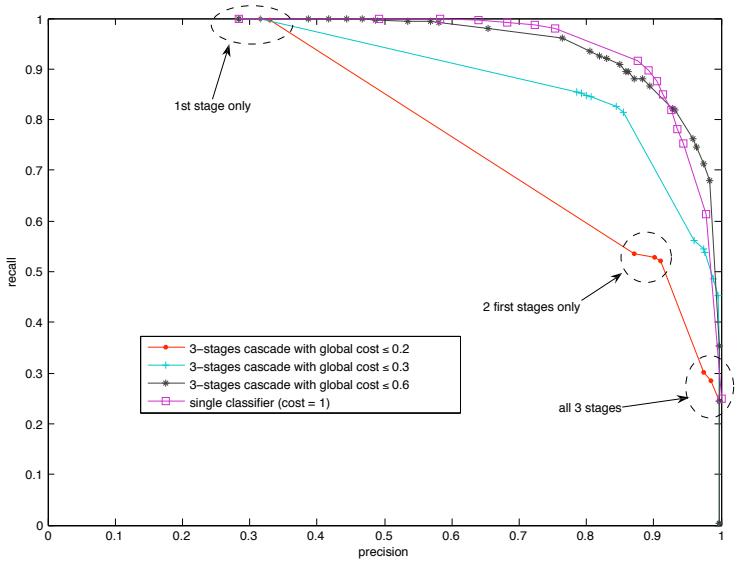


Fig. 5. Precision-recall curves. Comparison is made between different target costs.

for which the cost is 1. This is made possible thanks to the first stages that avoid many negative samples from being further decoded.

2) Stage skipping triggered by the optimization step: another observation is that on each curve (and particularly on "$cost \leq 0.2$" and "$cost \leq 0.3$"), we observe several "clusters" of operating points. Such cluster corresponds to some λ_{prec} values for which the optimization algorithm has decided to use only some of the available stages, as indicated on the figure. This is further illustrated in Fig. 6: for a given cost constraint, the global precision-recall curve is the convex-hull of 3 different curves, each corresponding to the use of a given subset of stages in the cascade. As we see, depending on the cost constraint and the importance of precision compared to recall (i.e. the value of λ_{prec}), it might be more efficient to use only 1 or 2 out of the 3 stages.

3) Better precision at low or moderate recall values: an interesting point to observe on Figure 5 is the better performances obtained by the cascade for moderate recall values (i.e. detection rates): when the cost-target is not too low (0.6 for instance), the cascade leads to a better precision than the single classifier, for recall values up to 0.83. This can be explained by the following arguments. When using a cascade, the whole classification problem is split in several successive smaller problems. This succession of sub-problems will focus on increasingly difficult samples.

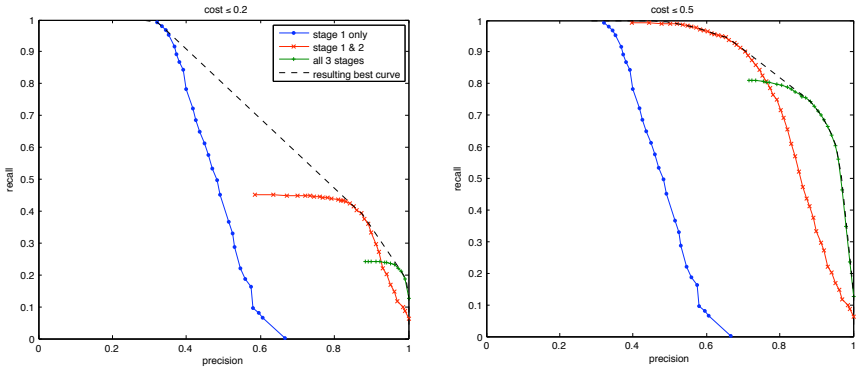


Fig. 6. Precision-recall convex-hull, sustaining the results obtained with different fixed number of stages.

Although not identical, this operation is conceptually similar to a boosting process [31] in which a classifier will be progressively forced to give more importance to ambiguous samples. It leads to stronger classifiers and explains that, as long as targeted detection rates are not too high, the cascade outperforms a single classifier that has to directly classify the whole set of samples with a single ensemble of independent random decision trees.

When targeted detection rates increase, each stage of the cascade will have to guarantee a very small amount of false negative errors. The first stages, that use a less discriminant features set than the last one, will therefore tend to choose smaller thresholds and let a greater number of samples reach the last stage. Indeed, this will increase the number of false positive errors but will guarantee a higher detection rate. This trend will progressively make the cascade more similar to a single classifier. We can therefore expect a decrease of the "boosting" effect described above and the cascade performances should get closer from the single classifier results.

4) Sub-optimality at high recall values: based on the explanation from the last paragraph, we should expect that if the cost-constraint is completely relaxed, the obtained cascade will outperform the single classifier at moderate detection rates and then get closer of (or even superimpose) the single classifier curve for higher rates. However, we observe that the curve crosses the single classifier curve and remains below it when the detection rate increases. This is due to the fact that the last stage has not been trained to classify all entering samples but only those having successfully passed the two first stages, set up with "normal" thresholds. If the cascade is subsequently degenerated to its last stage by the optimization step, it will remain sub-optimal in comparison to a single classifier whose training phase has been done on all samples. This sub-optimality should disappear

if we replace our post-learning optimization step with an iterative process that include the training phase in the loop, which would however drastically increase the computational load required to build the cascade.

VII. Conclusion

In this paper, we have investigated techniques able to perform classification and retrieval tasks based on JPEG 2000 code-streams. We have first studied how to extract relevant information from such code-streams in order to generate an image characterization that could subsequently be used in a classification process. We showed that, thanks to the various scalability levels inherently present in JPEG 2000, the quality of this characterization is directly related to the amount of data processed (and partially decoded) by the classifier. In particular, a new way to store the extracted feature values, called *integral volumes*, has been introduced and allows to very easily obtain relevant information on any area in the image. We then assessed the performances that can be expected when using JPEG 2000 features in various classification tasks based on random decision trees. The results obtained were similar to the best results available in the literature on raw images classification and categorization. This proves that staying in a JPEG 2000 framework not only avoid heavy transcoding operations and speed up image handling but allows also to get very good classification performances. Eventually, we combined several JPEG 2000 classifiers in a cascade suited for image retrieval operations. This cascade has been designed so as to draw benefit from the JPEG 2000 scalability: in particular, the amount of data that has to be extracted from a sample is directly related to the relevance of this sample. Optimization of this cascade takes into account the feature extraction cost and makes therefore a trade-off between complexity and performances. Results showed that performances similar to a single classifier are obtained, for a cost which is almost twice smaller.

Several interesting future research directions have already been identified on different aspects of the presented system. Among them, one could cite the study of a better way to combine information from various subbands in order to increase the robustness to orientation changes. The cascade also could be improved so as to get better results at high recall values. To achieve this, we plan to investigate unsupervised learning, which would allow to continuously learn and optimize the cascade. In this way, our goal is to build stages that are always trained with learning data as close as possible from the test data.

More globally, this work has to be seen as a step towards a user-centered online image retrieval

system. While experiments in this paper used offline computed samples sets, the goal is to be eventually able to interact with the user and to progressively learn what he is looking for.

Appendix A
Databases

- **PONCE**
  - *Description*: the Ponce Texture Database, introduced in [19], is a dataset of 640x480 grayscale images of 25 different textures with 40 images per class (1000 images in total). See Fig. 2(a) page 9 for a few samples.
  - *Protocol*: the learning set is made of 39 images from each of the 25 classes. The remaining 25 images compose the test set. The final error rate is the average of 40 error rates, the ith rate being obtained by using image i from each class for the test set.
- **ZUBUD**
  - *Description*: the ZuBud dataset [32] contains color images of 201 different buildings in Zürich. Each building in the training set is represented with five 640x480 images (1005 images in total). The test set is made of 115 images of size 320x240, covering a subset of the 201 different classes. The two sets were taken by two different cameras, in different seasons and under different weather conditions, leading to various illumination conditions. Partial occlusions and cluttered backgrounds, as well as scale and orientation changes, are often present in the images.
  - *Protocol*: as mentioned above, learning and test sets are provided online as separate sets. Therefore, we did not have to define a protocol to spli the images in two different sets.
- **ETH-80**
  - *Description*: the cogvis ETH-80 dataset is targeted specifically to the task of object categorization. It is made of 3280 color images (256x256) divided in 8 distinct categories (apples, pears, tomatoes, cows, dogs, horses, cups, cars). For each category, 10 different objects are provided. Each object is represented with 41 different images from viewpoints equally spaced over the upper viewing hemisphere.
  - *Protocol*: the protocol used is the same as the one used by Leibe and Schiele in [30], i.e. a leave-one-object-out cross-validation. The learning set is made of all views of 79 objects (which means a set of 3239 images) and the test is operated on all 41 views of the remaining object. Results are averaged over all 80 possible test objects.
- **COIL-100**
  - *Description*: the COIL-100 dataset [33] from the Columbia Object Image Library is made of 7200 color 128x128 images. There are 100 classes, each one representing a different 3D object. Each object is represented by 72 images at pose intervals of 5 degrees.
  - *Protocol*: two different protocols were used, like in [29]. *Protocol 1* takes the pose at 0° of each object as the learning set (100 images in total) and the remaining images as the test set (7100 images). *Protocol 2* takes 18 images of each object (two images of the same object being separated by 20°) as the learning set (1800 images) and the remaining ones as the test set (5400 images).

References

[1] J. F. Gantz and D. R. *et al.*, "An Updated Forecast of Worldwide Information Growth Through 2011," IDC white paper, March 2008.

[2] S. Livens, P. Scheunders, G. van de Wouwer, and D. Van Dyck, "Wavelets for texture analysis, an overview," *Image Processing and Its Applications, 1997., Sixth International Conference on*, vol. 2, 1997.

[3] J. Jiang, B. Guo, and S. Ipson, "Shape-based image retrieval for JPEG-2000 compressed image databases," *Multimedia Tools and Applications*, vol. 29, no. 2, pp. 93–108, 2006.

[4] A. Tabesh, A. Bilgin, K. Krishnan, and M. W. Marcellin, "Jpeg2000 and motion jpeg2000 content analysis using codestream length information," in *Proceedings of the Data Compression Conference (DCC'05)*, 2005.

[5] F. Crow, "Summed-area tables for texture mapping," *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pp. 207–212, 1984.

[6] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[7] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 36, no. 1, pp. 3–42, 2006.

[8] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification," in *Proceedings of the IEEE Intern. Conf. on Comp. Vision and Pattern Recog.*, vol. 1, June 2005, pp. 34–40.

[9] F. Fleuret and D. Geman, "Coarse-to-fine face detection," *International Journal of Computer Vision*, vol. 41, no. 1-2, p. 85, Jan 2001.

[10] J. Bi and S. e. Periaswamy, "Computer aided detection via asymmetric cascade of sparse hyperplane classifiers," *Proc. of the 12th ACM SIGKDD on Knowledge discovery and data mining*, pp. 837–844, 2006.

[11] A. Descampe, P. Vandergheynst, C. De Vleeschouwer, and B. Macq, "Coarse-to-fine textures retrieval in the JPEG 2000 compressed domain for fast browsing of large image databases," in *Inter. Workshop on Multimedia Content Repres., Classif. and Sec. (IWMRCS)*, September 2006.

[12] M. Boliek, C. Christopoulos, and E. Majani, "JPEG 2000 image core coding system (Part 1)," ISO/IEC JTC1/SC29 WG1, Tech. Rep., July 2001.

[13] D. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice.* Boston, MA, USA: Kluwer Academic, 2002.

[14] S. Mallat, *A Wavelet Tour of Signal Processing.* Academic Press, 1998, 577p.

[15] M. K. Mandal and C. Liu, "Efficient image indexing techniques in the JPEG2000 domain," *Journal of Electronic Imaging*, vol. 13, pp. 179–187, Jan. 2004.

[16] S. Mallat and W. Hwang, "Singularity detection and processing with wavelets," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 617–643, March 1992.

[17] M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance," *IEEE Trans. Image Process.*, vol. 11, no. 2, 2002.

[18] A. Teynor, W. Muller, and W. Kowarschick, "Compressed Domain Image Retrieval Using JPEG2000 and Gaussian Mixture Models," *Lecture Notes in Computer Science*, vol. 3736, p. 132, 2006.

[19] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, August 2005.

[20] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "CART: Classification and Regression Trees," *Wadsworth: Belmont, CA*, 1983.

[21] T. Dietterich and E. Kong, "Machine learning bias, statistical bias, and statistical variance of decision tree algorithms," *Dept. of Computer Science, Oregon State University Technical Report*, 1995.

[22] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[23] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 1, pp. 23–38, 1998.

[24] S. Brubaker, C. Wu, J. Sun, J. Mullin, M. D. Rehg, and M. James, "On the design of cascades of boosted ensembles for face detection," Georgia Institute of Technology, Tech. Rep., 2005.

[25] M. M. Dundar and J. Bi, "Joint Optimization of Cascaded Classifiers for Computer Aided Detection," in *Computer Vision and Pattern Recognition Conference*, 2007.

[26] H. Luo, "Optimization design of cascaded classifiers," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005.

[27] J. Davis, V. S. Costa, I. Ong, D. Page, and I. Dutra, "Using bayesian classifiers to combine rules," 3rd SIGKDD Workshop on Multi-Relational Data Mining (MRDM 2004). In conjunction with KDD 2004., 2004.

[28] A. Descampe, C. De Vleeschouwer, L. Jacques, and F. Marques, *How does digital cinema compress images?, in "Applied Signal Processing: A Matlab-based Proof of Concept".* Springer Verlag, 2008, ch. 10, pp. 361–410.

[29] J. Matas and S. Obdrzalek, "Object recognition methods based on transformation covariant features," in *Proc. 12th European Signal Processing Conference (EUSIPCO)*, 2004.

[30] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, 2003.

[31] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal Comput. System Sci.*, vol. 55, pp. 119–139, 1997.

[32] H. Shao, T. Svoboda, and L. Van Gool, "ZuBuD—Zurich Buildings Database for Image Based Recognition," *Technique report No. 260, Swiss Federal Institute of Technology*, 2003.

[33] H. Murase and S. Nayar, "Visual learning and recognition of 3-d objects from appearance," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.