Class-Specific Binary Correlograms for Object Recognition

Jaume Amores[1], Nicu Sebe[2], Petia Radeva[3]
[1] IMEDIA Research Group, INRIA, France
[2] Univ. of Amsterdam, The Netherlands
[3] Computer Vision Center, UAB, Spain

Abstract

This paper presents an efficient object-class recognition approach based on a new type of image descriptor: the Class-Specific Binary Correlogram (CSBC). In our representation, the image is described by a collection of CSBCs, where each one encodes the spatial distribution of class-specific features around a particular reference point. This representation is obtained by first performing an automatic selection of class-specific features from a vocabulary, and then extracting collections of binary correlograms that encode, at the same time, detected object parts and their spatial distribution around multiple points of the image. Our descriptors live in high-dimensional spaces (in the order of 10K dimensions), but they are very sparse. We show that efficient learning and matching procedures can be obtained for such a representation if we use, first, fast feature selection techniques specific for binary features, and then Boosting integrated with an appropriate Inverted File data organization. The proposed strategy works with weak supervision, outperforms state-of-the-art bag-of-feature methods, and it is more accurate and computationally more efficient than well-known geometrical-based methods, including our previous work on Generalized Correlograms (GCs) [1].

1 Introduction

In this paper we deal with the problem of Object-Class Recognition, which has received much attention recently [6, 14, 16, 15, 12]. The objective is to detect the presence or absence of objects from a desired category (for example 'car') in the images presented to the system. The problem is challenging due to the large variability of object appearance across instances of the category, added to the variability of pose and illumination, partial occlusion and clutter.

A fundamental issue is how relevant information is extracted from the image and how it is efficiently managed in order to first learn a model of the desired category, and then scan for instances of this model in new images. There is broad consensus upon the suitability of describing the local features and their spatial relation: local features are more robust against clutter than global signatures, and considering the mutual position of features significantly increases the distinctiveness of the representation. However, using spatial relations usually leads to much higher costs in the learning and matching procedure. The common approach is to use graph-base representations where local features and

spatial relations are described respectively by nodes in the graph and arcs between nodes. Despite its flexibility, the problem with such an approach is that learning and matching graph representations is known to be very expensive, even if we use fast optimization procedures [7, 4, 9].

Instead of representing these two types of information (local information about features and geometrical information about the mutual position of these features) by two separate entities (nodes and arcs), we proposed in [1] a Generalized Correlogram (GC) that encodes in the same feature vector the local information describing what are the local features in the image and, at the same time, the geometrical information describing the mutual position of these features. The advantage of such a representation is that, by simply comparing two feature vectors we take into account simultaneously the similarity of local features and their spatial distribution. This allows us to employ fast matching techniques that quickly consider the relevant information.

The Generalized Correlogram from our previous work [1] is a joint distribution of local and geometrical attributes, similar in spirit to previously defined correlograms in the literature [10, 3], but specially suitable for the object-class recognition task. It was defined to be generic in order to be used in retrieval of large databases, where the descriptors of database images were extracted off-line, i.e. regardless of the query the user was interested on. In this paper, we explore an alternative framework and introduce a Class-Specific Binary Correlogram (CSBC), which varies depending on the object category to be recognized. Along with it, we describe fast learning and matching procedures that permit to efficiently manage this type of description by exploiting its binary nature and its high degree of sparseness.

The rest of the paper is organized as follows: section 2 describes the image representation, section 3 describes the construction of model contexts in the learning the object class, section 4 describes fast procedures for learning our representation, section 5 describes the matching step with Inverted Files, section 6 shows results and section 7 concludes.

2 Image representation

The image is represented by a collection of Class-Specific Binary Correlograms (CSBC), each one describing the spatial distribution of class-specific features around a certain reference point of the image. For this purpose, the method: i) extracts a set of local features which convey relevant information about the object category, ii) detects the presence of these class-specific features in each image; and iii) extracts a set of correlograms, each one describing the mutual position of detected features relative to a certain reference point of the image. As we will see, our CSBC descriptors make use of a log-polar spatial quantization [3] that make them semi-local, so that CSBC focus on the local information around a certain reference, and incorporates the contextual properties (how the rest of detected object parts are distributed around the reference) in a smoothly decreasing degree of attention. Let us describe each step of the representation procedure.

2.1 Extraction of class-specific local features

Local features specific of the object category are gathered by using feature selection over a large pool of features extracted from training images. The extraction procedure is divided in the following stages: i) extraction of a large pool of local features, referred to as

"dictionary" in the bag-of-features paradigm [13, 21, 5] and ii) selection of a reduced set of class-specific features that convey relevant information about the object class.

2.1.1 Building a dictionary of local features

Let the training set T consist of N images $T = \{I_1, \ldots, I_N\}$, and let the i-th image $I_i \in T$ have an associated set of local descriptors (features) $L_i = \{\vec{l}_{i1}, \ldots, \vec{l}_{iK}\}$ extracted from I_i. These local descriptors can be extracted either from interest points detected in the image or by simply performing a random sampling in both position and scale. In our implementation we employ the interest point extractor of Kadir et al. [11], which obtains a set of points with maximum entropy in scale-space, and then take the K most salient points. Finally, the vocabulary V is built by gathering all the local descriptors extracted from each training image: $V = \cup_{i=1}^{N} L_i$.

2.1.2 Selection of class-specific features

Given a large pool of features V, we want to select a reduced set which conveys the maximum amount of information about our particular object category. Different mechanisms have been employed in the literature for this purpose. The standard one finds class-specific features by observing the frequency of occurrence of each feature from the vocabulary in each image from the training set. Intuitively, those features that are more frequently found in foreground images than in background ones will convey information about our object class. This approach is followed in the well-known bag-of-features paradigm [13, 21, 5], where each image I is described by a histogram h that counts the number of times that each feature from the vocabulary V is matched with some feature of I. A problem observed this approach is the sensitivity to the number of bins utilized in the histograms.

In this work we follow an alternative method. For each image I_i of the training set, we compute a vector of distances $\vec{d}_i = (d_1, d_2, \ldots, d_M)$ where the j-th element measures the distance $d(\vec{v}_j, I_i)$ between the j-th feature $\vec{v}_j$ of the vocabulary and the image I_i. This distance is defined as the minimum distance from $\vec{v}_j$ to all the features found in I_i: $d(\vec{v}_j, I_i) = \min_{\vec{l} \in I_i} \|\vec{v}_j - \vec{l}\|$. In other words, for each $\vec{v}_j$, we match it against the feature $\vec{l} \in I_i$ with minimum distance, and keep the distance score. Intuitively, if $\vec{v}_j$ is some characteristic feature of the object, most of the foreground images in the training set will have some local feature that is similar to $\vec{v}_j$, whereas background images will have features that are dissimilar. Therefore, distances to foregrounds $d(\vec{v}_j, F)$ will be generally smaller than distances to backgrounds $d(\vec{v}_j, B)$.

Once we represent each training image I_i by a distance vector $\vec{d}_i$, we can employ several techniques to select class-specific features. If we use maximum information, we will obtain a method very similar to the method used by employed by Ullman et al. [18] for selecting their class-specific image fragments. Alternatively, we can use Boosting with decision stumps in order to obtain a small set of informative features that maximally complement each other in the classification task. The latter has the advantage of being computationally much more efficient, and it is the method that we use in this work. This method was followed by Opelt et al. in [14], although they did not use it for feature selection, but directly for obtaining a classifier.

Selecting class-specific features with AdaBoost and decision stumps as weak classifiers is straightforward, once we have each training image I_i described by the distance

vector $\vec{d_i}$. We only need to call the standard AdaBoost procedure with the training pairs $(\vec{d_i}, y_i), i = 1, \ldots, N$ where y_i is the label of the image I_i. We used the AdaBoost with decision stumps algorithm described by Viola et al. in [19], we do not reproduce it here due to lack of space.

As a result, AdaBoost obtains an ensemble of decision stumps $f_1, \ldots, f_K$, where K is a parameter introduced to AdaBoost. The k-th decision stump $f_k(\vec{d})$ is the function:

$$f_k(\vec{d}) = \begin{cases} 1 & if \quad d_{i_k} < \theta_k \\ 0 & otherwise \end{cases},$$

where i_k and θ_k are parameters defining f_k. In our case, $\vec{d}$ is the vector of distances computed for some image I, and the element i_k of this vector, d_{i_k}, represents the distance of image I to the feature $\vec{v}_{i_k}$ from the vocabulary:$d_{i_k} = d(\vec{v}_{i_k}, I) = \min_{\vec{l} \in I} \|\vec{v}_k - \vec{l}\|$. Therefore, in our case the following definition is equivalent:

$$f_k(I) = \begin{cases} 1 & if \quad \exists \vec{l} \in I : \|\vec{l} - \vec{v}_{i_k}\| < \theta_k \\ 0 & otherwise \end{cases}$$

In other words, the function f_k is activated over the image I iff there is an image feature $\vec{l} \in I$ whose distance to the vocabulary feature $\vec{v}_{i_k}$ is lower than the threshold θ_k. In our setting, Boosting selects those features $\vec{v}_{i_k}$ that better separate positive and negative images respect to their distance to $\vec{v}_{i_k}$. In other words, Boosting selects those features $\vec{v}_{i_k}$ that match (i.e. have a small distance) with some feature in every foreground image and that, at the same time, do not match (i.e. have a big distance) with all the features of background images, which is exactly what we need, and additionally it outputs the matching thresholds θ_k found to be statistically more consistent.

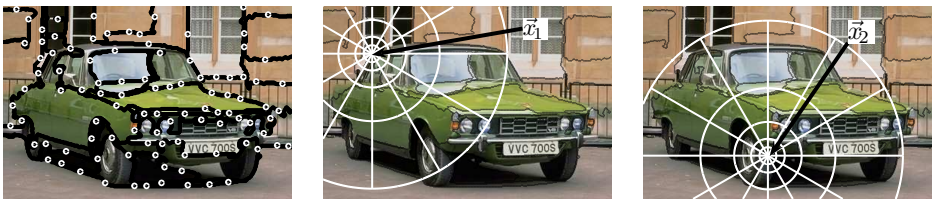
Finally, let us express as the output of this step as the set $C = \{\vec{c}_k\}_{k=1}^K$ of features, so that $\vec{c}_k = \vec{v}_{i_k}, k = 1, \ldots, K$. We will refer to features in $\vec{c} \in C$ as *class features*, because they convey information about our object class. Associated with C, we obtain a set of thresholds $\Theta = \{\theta_k\}_{k=1}^K$, where θ_k defines the matching threshold of the class feature $\vec{c}_k$, i.e. $\vec{c}_k$ matches with all the image features from I with distance to $\vec{c}_k$ lower than θ_k.

2.2 Extraction of Class-Specific Binary Correlograms

In our implementation, given image I we extract a dense set of image local descriptors sampled every 4 pixels in 7 different scales that range from 0.5 to 2 times the size of the image, with linear increments. We use a very basic type of local descriptors reported in [6]: local gray-level windows re-scaled to size 11×11 and projected by PCA to 15 dimensions, parameters used in [6]. Much better results can be obtained with more sophisticated descriptors, but our interest in this work was to measure the relative performance compared to previous methods with the same type of local descriptors.

Given the image I, let the class feature $\vec{c}_k \in C$ match with the set of image features $M_k = \{\vec{m}_j^k\}_{j=1}^{N_k}$ from the image, and let us define $P_k = \{\vec{p}_j^k\}_{j=1}^{N_k} \subset R^2$ to be the spatial positions from which these matching features are extracted, i.e. $\vec{p}_j^k \in P_k$ is the spatial position of matching feature $\vec{m}_j^k \in M_k$.

Let $X = \{\vec{x}_i\}_{i=1}^R$ be a set of reference points sampled from the image I. For each reference point $\vec{x}_i \in X$ we extract a Class-Specific Binary Correlogram h_i as follows. Let the spatial relation $(\vec{x}_i - \vec{p}_j)$ from the reference to any point $\vec{p}_j$ be expressed in polar



(a) (b) (c)

Figure 1: (a) Sampled set of points taken as reference. (b)-(c) Log-polar spatial quantization of our descriptor given two different references $\vec{x}_1$, $\vec{x}_2$.

coordinates $(\alpha_{ij}, r_{ij}) : \alpha_{ij} = (\widehat{\vec{x}_i - \vec{p}_j})$, $r_{ij} = \|\vec{x}_i - \vec{p}_j\|$. The angle and radius are quantized into n_α and n_r bins respectively. Let A_u be the u-th bin of angles, $u = 1, \ldots, n_\alpha$, and let R_v be the v-th bin of radius, $v = 1, \ldots, n_r$. We use the same log-polar spatial quantization as used by Belongie et al. for their Shape Context descriptor in [3], see fig. 1(b)-(c).

The class-specific correlogram h_i associated with $\vec{x}_i$ measures the distribution of points $\vec{p}_j^k$ according to their position relative to the reference $\vec{x}_i$ and the index k of the class feature. The distribution is estimated by the histogram:

$$h_i(u,v,k) = \#\{\vec{p}_j^k \in P_k : (\widehat{\vec{x}_i - \vec{p}_j^k}) \in A_u, \|\vec{x}_i - \vec{p}_j^k\| \in R_v\}, \quad (1)$$

Note that index k provides local information, as it indicates the class-specific local feature that is detected at position $\vec{p}_j^k$. Therefore, h_i measures a joint distribution of geometrical properties (the quantized spatial relations to the reference) and local properties (the type of matching feature). Instead of considering h_i, we binarize it to obtain a binary correlogram. This permits to compact considerably the correlogram, and allow us to employ fast techniques based on binary features. Also, in practice we did not observe a significant difference in performance, although a systematic comparison was not performed. In the end, we obtain a CSBC $h_i(u,v,k)$ that flags those spatial bins (u,v) where class feature $\vec{c}_k$ is detected.

As said before, the angle and radius are quantized using the log-polar quantization of [3] (see Figs. 1(b)-(c)), which makes the descriptor more sensitive to local context. The angle is quantized into $n_\alpha = 12$ bins, and the *logarithm* of the radius is quantized into $n_r = 5$ bins. If we use K class features, the CSBC h_i has $12 \times 5 \times K$ dimensions. For example, for $K = 250$ class features, we obtain 15000 dimensions. However, the descriptor does not occupy much space: first, it can be represented in a bit-wise manner, i.e. if we use integers of 32 bits the CSBC occupies only 15000/32 = 469 integers. Further, most of the elements are 0, so that we can use a sparse format which further reduces the space to a few integers.

Reference points in X are sampled from contour points extracted from the image, and we keep the R points with maximum distance to each other, so as to cover the image from every possible angle, a more detailed explanation can be found in [1], for an illustration see fig. 1(a).

In the final image representation we use a multi-scale representation, and express the image description as:

$$\mathscr{H} = \{H_s\}_{s=1}^{S}$$
$$H_s = \{\vec{h}_{is}\}_{i=1}^{R}$$

where H_s is a set of CSBCs scaled according to the s-th scale, and S is the total number of scales. CSBCs are scaled by normalizing the radius of the difference vectors expressed in polar coordinates.

3 Building a collection of model contexts

As we saw, images are represented by collections of correlograms, where each one represents a particular *image context*, i.e., how different types of local features are spatially distributed around a particular reference point. Given this representation, we also express the learned model as a collection of parametric *model contexts*: $\Omega = \{\langle \omega_c, \vec{\varphi}_c \rangle\}_{c=1}^{C}$, where ω_c identifies the c-th model context and is associated with vector of parameters $\vec{\varphi}_c$.

In order to build such a model, we have to match homologous image contexts across images of our training set. Therefore, each model context ω_c is learned by using a training set $\mathscr{T}_c$ of CSBC descriptors matched across the different training images. In order to explain how the matching is performed, let us first explain how an image is recognized once we have learned the model.

3.1 Recognition

Suppose that we have learned a model $\Omega = \{\langle \omega_c, \vec{\varphi}_c \rangle\}_{c=1}^{C}$, and we get a new image I that we want to evaluate, i.e., decide whether or not it contains an object that is an instance of the model Ω. Assume for now that descriptors are only extracted at one scale, so that the image I is represented by only one set $H = \{\vec{h}_i\}_{i=1}^{R}$.

Let $l(\vec{h}|\omega_c) \in [0,1]$ be the likelihood that the contextual descriptor $\vec{h}$ represents the model context ω_c. This likelihood is based on learned parameters $\vec{\varphi}_c$, and we use Boosting as explained in section 4. Let $L(H|\omega_c) \in [0,1]$ be the likelihood that *any* contextual descriptor in H represents ω_c. For computing this likelihood we use the maximum: $L(H|\omega_c) = \max_{\vec{h}_i \in H} l(\vec{h}_i|\omega_c)$. This can be regarded as matching the model context ω_c with the contextual descriptor $\vec{h}_{i^*}$ whose likelihood is maximum. We express this as $M(H|\omega_c) = \vec{h}_{i^*}$, where $M(H|\omega_c) = \vec{h}_{i^*} = \arg\max_{\vec{h}_i \in H} l(\vec{h}_i|\omega_c)$. Let $\mathbf{L}(H|\Omega) \in [0,1]$ be the likelihood that H represents the object according to the evidence provided by all the model contexts $\{\omega_c\}_{c=1}^{C}$ of our model constellation. As we want all the model contexts to contribute to this classification score, we use as combination rule the sum of likelihoods, with equal weight for each model context: $\mathbf{L}(H|\Omega) = \frac{1}{C}\sum_{c=1}^{C} L(H|\omega_c)$.

Consider now multiple scales $\mathscr{H} = \{H_s\}_{s=1}^{S}$ for image I. Let $\mathscr{L}(\mathscr{H}|\Omega)$ be the probability that *any* of the scaled representations $H_s \in \mathscr{H}$ of image I contains our object. This is computed again by using the maximum $\mathscr{L}(\mathscr{H}|\Omega) = \max_{H_s \in \mathscr{H}} \mathbf{L}(H_s|\Omega)$. Again, this can be regarded as matching the model object Ω with some scaled representation H_{s^*}, which is expressed as: $\mathscr{M}(\mathscr{H}|\Omega) = H_{s^*} = \arg\max_{H_s \in \mathscr{H}} \mathbf{L}(H_s|\Omega)$. The described procedure is similar to multi-scale Chamfer matching [2], using learned likelihoods instead of distances. This matching can be efficiently performed if we exploit the sparseness of our representation by means of inverted files, which is explained in section 5.

3.2 Matching with low supervision

As explained above, before learning the model we match homologous contexts in the training set. Our procedure consists of two stages. In the first stage, we apply the registration procedure of [3] to a small set of manually segmented images. As a result of

registration, we obtain sets of homologous contexts across a small number of images, and we can learn an initial model Ω'. Then, we use this model and the matching explained in the recognition section (functions $M(H|\omega'_c)$ and $\mathscr{M}(\mathscr{H}|\Omega')$) to obtain homologous descriptors for the rest of the training set. Basically, given a non-segmented image I from the training set, we match every model context $\omega'_c \in \Omega'$ with the descriptor of image I that maximizes the likelihood of representing ω'_c, and we use the scale that has the highest likelihood according to all the model contexts $\{\omega'_c\}_{c=1}^{C}$. After this matching, we can gather the complete sets of homologous contexts for all the training set, and learn the final model $\Omega = \{\omega_c\}_{c=1}^{C}$. We refer to [1] for a more detailed description of the procedure.

In order to learn the initial model Ω', we use Generalized Correlograms (GCs) with quantized local descriptors that represent the direction of edges, as done in previous work [1]. The motivation is that the first matching produces small *training sets* of homologous contexts and it is better to utilize GCs that have a small number of dimensions. In order to obtain the final model Ω, we use as contextual descriptors a concatenation of the previous GCs and the Class-Specific Binary Correlograms (CSBC) described in previous section 2.

4 Learning a compact model by fast binary feature selection and Boosting

We want to obtain a final model that is compact, in the sense that it is based on few dimensions of the large feature space where CSBCs live. For this purpose we propose to use a feature selection algorithm especially efficient in binary spaces, based on maximizing the conditional mutual information between selected features and the class label, and introduced by Fleuret in [8]. This algorithm allows to efficiently perform feature selection in very large binary spaces, and obtains a small set of dimensions that convey a high amount of information about our object category. In our case, each dimension flags the presence of some class-specific local feature located in some relative spatial position. Therefore, we obtain a compact model characterizing the relevant local parts of the object and their characteristic mutual position.

The Conditional Mutual Information Maximization (CMIM) [8] works by selecting those features that maximize their mutual information with the object class, conditional to any feature already picked. It ensures the selection of features which are both individually informative and two-by-two weakly dependant [8]. After efficiently obtaining a small set of class-informative features by CMIM, we use Boosting with decision stumps to learn the final parametric contexts $\vec{\varphi}_c$ of section 3. We use the Boosting implementation described by Viola and Jones in [19], which can be easily adapted to work with sparse data, as in our case, in order to efficiently learn the model, we refer to [1] for further detail. Further, Boosting with weak classifiers allows to easily and efficiently work with Inverted Files in order to match the model with new images. This is explained below.

5 Efficient matching with Inverted Files

If we use the Chamfer matching algorithm described in 3.1, and we use Boosting with decision stumps, we can easily adapt the matching procedure to efficiently work with Inverted Files (IF) from the information retrieval community [17], as we will briefly explain in this section (for more details we refer to [1]). We describe here a procedure valid for

Method	Motorbike	Plane	Car rear	Face	Cat	Leaf
Others, Gray	92.5% [6]	90.2% [6]	90.3% [6]	**96.4**% [6]	90.0% [6]	84.0% [20]
GC [1], Gray	**96.7**%	92.5%	**95.8**%	95.2%	**92.8**%	**98.0%**
GC [1], Col.	94.2%	94.4%	95.8%	90.5%	86.1%	96.3%%
CSBC, 250 c.s., 2000 d., Col.	**98.3**%	**98.7**%	**99.6**%	**95.7**%	**89.0**%	**100**%
CSBC, 25 c.s., 250 d., Col.	97.5%	96.4%	98.7%	94.4%	80.1%	95.7%

Table 1: ROC equal error rate in CALTECH.

both binary CSBCs and non-binary GCs, because we use GCs in the first matching step of 3.2. Each decision stump from Boosting is defined by a threshold θ, a selected dimension k and a weight α. The stump evaluates as positive those descriptors whose value in k-th dimension is lower than θ. For each entry (dimension) in the IF the *negative* values are stored in ascending order, so that the virtual zeros are put at the end of the list (when training the classifier we also use negative values). From the first descriptor found, we visit all the descriptors until the beginning of the list and we add α to the likelihood of these descriptors. Let p_k be the fraction of descriptors in the list whose value is lower than θ: $0 \leq p_k \leq 1$. The cost of evaluating the weak classifier is $O(\log n_k + p_k n_k)$. Note that we can evaluate all the weak classifiers that share the same dimension k in only one pass along the IF list of this dimension.

6 Results

In order to test the performance of the method, we used the CALTECH database, which has become a standard database for object-class recognition. For a description of the database and image examples we refer to [6], we do not show them here due to lack of space. Most of the object categories have instances under the same bi-dimensional arrangement, except for the cat category. Each category has roughly 800 images of different objects of this category. From the positive training set, only 10 images are manually segmented and the rest are left unsegmented. Two background sets are used: the original gray-level backgrounds [6] and color bakgrounds. The latter was found to be harder to classify than the gray-level background set.

In table 1 we compare the performance against well-known geometry-based methods and against our previous work using Generalized Correlograms [1]. The first two rows are scores with Gray backgrounds and the three last rows use more difficult Color backgrounds. 250 c.s + 2000 d. means that we use 250 class-specific features and we select 2000 dimensions for obtaining the final model. We used 2000 dimensions for obtaining the highest scores, although a much smaller number (250) obtain similar scores. As we can see, CSBCs significantly outperform GCs with color bacgkrounds, and GCs in turn outperform the well-known benchmark of Fergus et al. [6] with gray-levels. We did not test our CSBCs with gray-level backgrounds. In table 2 we compare our method against well-known methods that also use dictionaries of class-specific features but without integrating geometry. We implemented the bag-of-feature methods by using histograms, as in [13]. For fair comparison, we used the same setting (local descriptors, vocabulary extraction, etc.) as the one used for our CSBCs. We also implemented the version of Opelt et al. [14] by using the same setting as in our method. We can see that using Class-Specific Binary Correlgrams we outperform methods based only on class-specific local features.

Method	Motorbike	Plane	Car rear	Face	Cat	Leaf
Bag-of-features, 250 c.s.	90.8%	95.0%	98.9%	95.0%	91.0%	96.7%
dist. vec. [14] 250 c.s.	96.0%	95.8%	99.1%	95.2%	**92.0**%	96.7%
CSBC, 250 c.s., 2000 dim.	**98.3**%	**98.7**%	**99.6**%	**95.7**%	89.0%	**100**%
dist. vec. [14] 25 c.s.	90.0%	95.0%	95.6%	91.7%	**88.0**%	94.6%
CSBC, 25 c.s., 250 dim.	**97.5**%	**96.4**%	**98.7**%	**94.4**%	80.1%	**95.7**%

Table 2: ROC equal error rate in CALTECH. Comparison with methods based on dictionaries of class-specific features.

Using CSBCs can be seen as a post-processing step after performing a method that obtains class-specific features, such as [13, 14]. The cost of extracting the multi-scale CSBCs in a Pentium IV at 3.0GHz was around 0.7 seconds per image, with code implemented in Matlab and some subroutines in C. The cost of classifying an image with 2000 selected features and inverted file was around 1/3 seconds in a Pentium IV at 3.0 GHz. Note that a smaller number of dimensions is enough for obtaining almost the same results: with 500 dimensions the average difference in accuracy is 0.62%, and we classify an image in $1/15$ seconds. Learning CSBCs of 15000 dimensions and 2000 weak classifiers takes about 80 minutes, whereas using 500 dimensions we spend only 20 minutes approximately. These costs are much smaller than using fast graph-based methods such as [6], were the authors spend around 24 hours (in a 2.0 GHz processor). Using our GCs in [1] and a 2.4 GHz we spent 1 hour and 30 minutes for the whole learning algorithm and 250 selected features.

CSBCs share the same robustness as GCs, evaluated in [1]. Notably they are invariant to translation, robust against scaling in our case (we use a multi-scale approach) and robust against small rotations. This is in contrast with recently proposed fast geometry-based methods that are not robust against pose variation.

7 Conclusions

We showed how to efficiently consider spatial relations by using Class-Specific Binary Correlograms. For this purpose, we outlined an architecture were all the components are designed so as to efficiently learn and match this type of descriptors, i.e. by exploiting their sparseness and their binary nature. Notably, we saw that the proposed method outperforms recent methods based on dictionaries of class-specific features. In this sense, using CSBCs can be considered as an efficient post-processing step after after obtaining a set of class-specific features used in many recent works [13, 14, 12, 16]. We saw that this step can be done without high cost and significantly increments performance. As future work we will test it in the CALTECH-101 database and will use rich features as [12, 16].

References

[1] J. Amores, N. Sebe, and P. Radeva. Context-based object-class recognition and retrieval by generalized correlograms. *PAMI*. IN PRESS (on-line at IEEE web site).

[2] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. Int. Joint Conf. Artificial Intelligence*, pages 659–663, 1977.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(24):509–522, April 2002.

[4] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *IEEE CVPR*, 2005.

[5] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 2005.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. CVPR*, 2003.

[7] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proc. CVPR*, pages 380–387, 2005.

[8] F. Fleuret. Fast binary feature selection with conditional mutual information. *Machine Learning*, 5:1531–1555, 2004.

[9] P. Hong and T. S. Huang. Spatial pattern discovery by learning a probabilistic parametric model from multiple attributed relational graphs. *Journal of Discrete Applied Mathematics*, 139(1-3):113–135, April 2003.

[10] J. Huang, S. R. Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Spatial color indexing and applications. *IJCV*, 35(3):245–268, 1999.

[11] T. Kadir and M. Brady. Scale, saliency and image description. *IJCV*, 2001.

[12] J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, volume 1, pages 11–18, 2006.

[13] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proc. ECCV*, 2006.

[14] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3):416–431, 2006.

[15] S. Lazebnik C. Schmid and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[16] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 29(3):411–426, 2007.

[17] D. Squire, W. Muller, H. Muller, and T. Pun. Content-based query of image databases: Inspirations from text retrieval. *Pattern Recognition Letters*, 2000.

[18] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, 2002.

[19] P. Viola and M. J. Jones. Robust-real time face detection. *IJCV*, 2004.

[20] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *CVPR*, pages 101–108, 2000.

[21] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. ICCV*, 2005.