Fast and Scalable Image Retrieval Using Predictive Clustering Trees

Ivica Dimitrovski[1], Dragi Kocev[2], Suzana Loskovska[1], and Sašo Džeroski[2]

[1] Faculty of Computer Science and Engineering, University of Ss Cyril and Methodius
Rugjer Boshkovikj 16, 1000 Skopje, Macedonia
[2] Department of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{ivica.dimitrovski,suzana.loshkovska}@finki.ukim.mk,
{Dragi.Kocev,Saso.Dzeroski}@ijs.si

Abstract. The recent overwhelming increase in the amount of available visual information, especially digital images,has brought up a pressing need to develop efficient and accurate systems for image retrieval. State-of-the-art systems for image retrieval use the bag-of-visual-words representation of the images. However, the computational bottleneck in all such systems is the construction of the visual vocabulary (i.e., how to obtain the visual words). This is typically performed by clustering hundreds of thousands or millions of local descriptors, where the resulting clusters correspond to visual words. Each image is then represented by a histogram of the distribution of its local descriptors throughout the vocabulary. The major issue in the retrieval systems is that by increasing the sizes of the image databases, the number of local descriptors to be clustered increases rapidly: Thus, using conventional clustering techniques is infeasible. Considering this, we propose to construct the visual codebook by using predictive clustering trees, which are very efficient and have good performance. Moreover, to increase the stability of the model, we propose to use random forests of predictive clustering trees. We evaluate the proposed method on a benchmark database of a million images and compare it to other state-of-the-art methods. The results reveal that the proposed method produces a visual vocabulary with superior discriminative power and thus better retrieval performance.

Keywords: image retrieval, visual vocabulary construction, predictive clustering.

1 Introduction

An ever increasing amount of visual information is becoming available in various digital archives. For instance, the widely used social web sites such as FACEBOOK[1] and FLICKR[2] store several billions images for their users. The improvement of digital cameras and user interfaces for upload of images will further

[1] Facebook©– http://www.facebook.com
[2] Flickr from Yahoo!©– http://www.flickr.com

J. Fürnkranz, E. Hüllermeier, and T. Higuchi (Eds.): DS 2013, LNAI 8140, pp. 33–48, 2013.

increase the amount of available images. The value of the information obtained from an image depends on how easily it can be found, retrieved, accessed, filtered and managed. Considering this, the development of systems for efficient archiving, browsing and searching images is a necessity. Such systems are being developed within the research area of image retrieval.

Image retrieval is an inter-disciplinary research area that cross-fertilizes the following research areas: multimedia research, information retrieval, machine learning, computer vision, and human-computer interaction. The methods for image retrieval can be categorized into two categories [1]: text-based image retrieval (TBIR) and content-based image retrieval (CBIR). The former group of methods require some meta-data for each image in textual format (i.e., image tags) and then the retrieval is performed by providing textual queries. These methods perform well and are efficient as long as the images are correctly tagged. However, these methods have two serious limitations: a large amount of human labour for manual annotation (which is even more exacerbated in the case of large image databases) and the inaccuracy from the subjectivity of the human annotators. To alleviate these limitations, CBIR methods were introduced. They describe the images by their visual content, such as color, texture, shapes and local descriptors. The CBIR methods heavily rely on extracting appropriate image descriptors and a good similarity measure between images.

In this work, we focus on developing a method for efficient CBIR in large scale image databases. More specifically, we are concerned with developing a method for particular object retrieval from a large scale database which will retrieve all images that contain the specific query object. In other words, we are interested to associate two images based on the objects they contain and not based on the entire images. For example, if the query object is a specific model of a BMW car, then the method should retrieve the images containing that specific model of a BMW and not other models of a BMW or any other type of car.

These methods can be readily applied in several practically relevant domains [2]. To begin with, they can be used for creation of a web-scale visual search engine where the query will be a visual object. Second, they can be used for searching through personal image databases (e.g., select the photos that contain an image of the Eiffel Tower). Next, performing product search will strongly benefit from such systems: The user can take a photo of a given product, perform a search on the web and compare its prices from the given store to the price in the on-line shops. Furthermore, these methods will facilitate automatic tagging of images upload on social media, such as Flickr and Facebook. Finally, these methods can be used for augmented reality and creation of visual guides for museums and art galleries. For instance, a user can take a photo of a given sculpture and look for info on the web for the given sculpture.

Several systems for particular object retrieval have been proposed in the literature [3,2,4]. These systems are inspired from the text retrieval systems using the analogy between bag-of-words and bag-of-visual-words representation [5]. They consist of three phases: creation of visual vocabulary, image description and similarity definition. The creation of the visual vocabulary starts with detection of

interesting points in the images. From these points, then, local invariant descriptors are extracted. Finally, the visual codebook is obtained by clustering the large set of descriptors obtained from all of the images. The resulting clusters represent the visual word, while all the visual words comprise the visual dictionary. The image description phase consists of assigning all of the local image descriptors to the visual words from the visual dictionary. Each image is then described with a high-dimensional histogram and each component from the histogram is the number of descriptors that are assigned to a given visual word. Finally, the images are ranked using term frequency inverse document frequency ($tf - idf$) scores which discount the influence of visual words which occur in many images. The search is then performed efficiently using a fast and tree-based inverted index structure [6].

The systems for particular object retrieval face several challenges [2]. To begin with, the changes in the lighting, image scale and rotation can hurt the performance of the retrieval systems. Second, viewpoint changes can make previously unseen parts of the object visible and also may include obstructions which will cover parts of the object. Finally, the systems need to be scalable with respect to the size of the image database, while preserving the retrieval accuracy. Moreover, the construction of the visual vocabulary should be also performed more efficiently.

In this paper, we present a novel method for fast and efficient construction of the visual codebook for particular object retrieval. The proposed method is based on the predictive clustering framework [7] which unifies predictive modelling and clustering through methods that partition the instances into subsets, such as decision trees and decision rules. The task of predictive clustering is to identify clusters of instances that are close to each other both in the target and in the descriptive space. In this work, we use the predictive clustering trees (PCTs) to construct the visual vocabulary. More specifically, we use PCTs for predicting multiple targets in which the descriptive attributes are also considered as clustering/target attributes.

Using a single PCT is a fast and efficient approach to the construction of visual codebooks. However, PCTs (and decision trees in general) are unstable, i.e., can change substantially for small changes in the data. To further improve the discriminative power and the robustness of our approach, we are using a small ensemble (random forest) of PCTs (as suggested in [8]). This produces several visual codebooks (each codebook corresponding to a single PCT from the ensemble). The overall visual codebook is then obtained by concatenating the visual codebooks from the single PCTs.

The remainder of this paper is organized as follows. Section 2 briefly presents the related state-of-the-art methods for particular object retrieval. The predictive clustering framework is described in Section 3. Section 4 gives the proposed method for codebook construction for particular object retrieval. Section 5 outlines the experimental design, while Section 6 presents the results from the experimental evaluation. Finally, the conclusions and a summary are given in Section 7.

2 Related Work

In this section, we briefly present the most widely used and state-of-the-art methods for CBIR. Many studies have shown that the bag of visual words approach exhibits a high retrieval performance for object retrieval given a query image of some particular object [9,10]. A crucial step in the bag of visual words approach is the codebook construction. The best results are achieved by using large codebook that contains one million or even more entries/visual words, which requires clustering tens or even hundreds of millions of high-dimensional feature descriptors into one million or more clusters. The methods for CBIR mainly differ in the process of the visual codebook construction.

The visual codebook is typically constructed by applying k-means clustering to the local descriptors (e.g., Scale Invariant Feature Transform descriptors) extracted from the images [11,12]. The resulting clusters are actually the visual words. Although this method works very well for smaller databases (and consequently small number of descriptors), it has very serious limitations when applied to the problem of large scale object retrieval [13]. It works with small visual codebooks, i.e., with only thousands of visual words, while many datasets may have tens of thousands of visual words.

The hierarchical k-means (HKM) approach [3] addresses the issue of small visual codebook. HKM performs the clustering in a hierarchical manner with a predefined number of levels (n). At the first level, the descriptor space is clustered into k_1 clusters, then at the second level, each of the k_1 clusters is re-clustered into k_2 clusters, and so on, until level n. The final visual codebook then consists of $k_1 \cdot k_2 \cdot ... \cdot k_n$ visual words. However, a major drawback of this method is that it is not a priori clear how many levels to use and how to choose the appropriate values for $k_1 ..., k_n$.

Philbin [2] proposed the approximate k-means (AKM) algorithm. In AKM, the exact nearest neighbor search is replaced with approximate nearest neighbor search in the assignment step when searching for the nearest cluster center for each point. In particular, the current cluster centers in each k-means iteration are organized by a forest of $k - d$ trees to perform an accelerated approximate nearest neighbor search.

Most closely related to our approach is the CBIR method based on indexing random sub-windows (extracted from the images) with extremely randomized trees [14]. These sub-windows are sampled from the images at random positions and random sizes. The sampled sub-windows are resized using bilinear interpolation to size 16×16. Each of the sub-windows is then described with the HSV values (thus resulting in a 768 feature vectors). Next, these feature vectors are used to construct extremely randomized tree: each node split is selected randomly; thus, these trees are constructed in an unsupervised manner. These trees are then used as search structures for the retrieval phase. Furthermore, the extremely randomized trees method can be used to construct ensembles thus further improve their retrieval performance [8].

Uijlings et al. [15] have performed experimental comparisons of visual dictionaries constructed using k-means and tree-based approaches. The main

conclusions from their study is that the tree-based approaches are more efficient than approaches based on k-means. However, the improvement of the computational efficiency comes with the price of decreasing the discriminative power of the vocabulary.

In this paper, we propose a method for visual codebook construction using the predictive clustering framework. More specifically, we propose to construct the visual codebook using PCTs and random forests of PCTs. There are two major differences between the method proposed here and the one proposed by Mareé et al. [14]. First, the former is used in the vocabulary construction phase and then it employs the $tf - idf$ scores and inverted index for the retrieval process, while the latter method tries to emulate the complete process. Second, the selection of splits in the former is performed with an informed approach, while in the latter method this is done randomly. Namely, the split selection in the proposed method is performed by considering the compactness of the produced clusters. This split selection is the main reason for obtaining a visual vocabulary with high discriminative power.

3 Predictive Clustering

In this section, we first outline the predictive clustering framework, which is the foundation of our codebook generation approach. We then briefly describe the predictive clustering trees for predicting multiple targets. Finally, we present the method for construction of random forests of predictive clustering trees.

3.1 Predictive Clustering Framework

The notion of *predictive clustering* was first introduced by Blockeel [7]. The predictive clustering framework unifies two machine learning techniques, predictive modelling and clustering, usually viewed as completely different. The connection between these techniques is made by machine learning methods that partition the instances into subsets, such as decision trees and decision rules. These methods can be considered both as predictive and as clustering methods.

The task of predictive clustering is to identify clusters of instances that are close to each other both in the target and in the descriptive space. Figure 1 illustrates the tasks of predictive modelling (Figure 1(a)), clustering (Figure 1(b)) and predictive clustering (Figure 1(c)). Note that Figure 1 presents the target and the descriptive space as one-dimensional axes for easier visual interpretation, but they are usually of higher dimensionality.

The clusters that were obtained using the target space only (Figure 1(a)) are homogeneous in the target space (the target variables of the instances belonging to the same cluster have similar values). On the other hand, the clusters obtained using the descriptive space only (Figure 1(b)) are homogeneous in the descriptive space (the descriptive variables of the instances belonging to the same cluster have similar values). The predictive clustering combines these two and produces clusters that are homogeneous both in the target and in the descriptive space (Figure 1(c)).

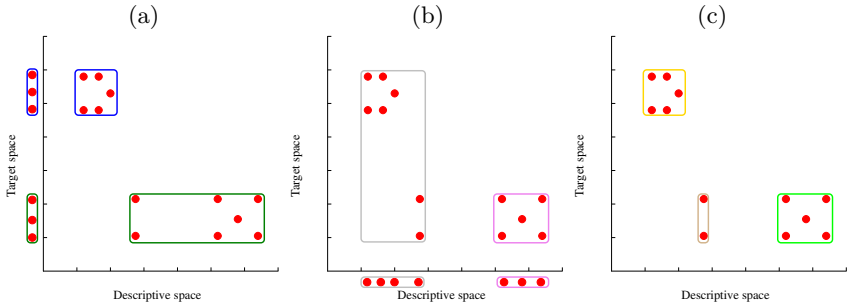

Fig. 1. An illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and the descriptive space. Figure adapted from [7].

In this work, we use a specific setting from the predictive clustering framework where the descriptive space is equal to the target space, i.e., the target variables are used to provide descriptions for the obtained clusters. This focuses the predictive clustering setting more on the task of clustering. This approach has two major advantages over classical clustering (such as k-means). First, we obtain the clusters much more efficiently as compared to standard clustering algorithms. Second, there are cluster descriptions for each of the clusters. The cluster description is the conjunction of the tests starting from the root node of the tree then following the path to the leaf (or the conditions used in a predictive clustering rule). This also improves the efficiency when new examples need to be projected into the clusters.

3.2 PCTs for Multiple Continuous Variables

The predictive clustering framework is implemented using decision trees (called predictive clustering trees - PCTs) and decision rules (called predictive clustering rules) as predictive models. The predictive clustering framework sees a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. PCTs can be induced with a standard *top-down induction of decision trees* (TDIDT) algorithm [16]. The algorithm is presented in Table 1. It takes as input a set of examples (E) and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction in variance caused by partitioning ($\mathcal{P}$) the instances (see line 4 of BestTest procedure in Table 1). By maximizing the variance reduction the cluster homogeneity is maximized and it improves the predictive performance. If no acceptable test can be found (see line 6), that is, if the test does not significantly reduces the variance, then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf.

The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the

Table 1. The top-down induction algorithm for PCTs

procedure PCT(E) **returns** tree	**procedure** BestTest(E)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$	1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **if** $t^* \neq none$ **then**	2: **for each** possible test t **do**
3: **for each** $E_i \in \mathcal{P}^*$ **do**	3: $\mathcal{P} =$ partition induced by t on E
4: $tree_i = \text{PCT}(E_i)$	4: $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{\lvert E_i \rvert}{\lvert E \rvert} Var(E_i)$
5: **return** node(t^*, $\bigcup_i \{tree_i\}$)	5: **if** $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ **then**
6: **else**	6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **return** leaf(Prototype(E))	7: **return** $(t^*, h^*, \mathcal{P}^*)$

prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task. So far, the predictive clustering framework has been used for the prediction of multiple continuous variables, prediction of multiple discrete variables, hierarchical multi-label classification (HMC) and prediction of time series [17]. The predictive clustering framework is implemented in the CLUS system[3].

The variance and prototype functions of PCTs for predicting multiple continuous variables are instantiated as follows. The variance is calculated as the sum of the variances of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Var(Y_i)$. The variances of the target variables are normalized, so that each target variable contributes equally to the overall variance. This is due to the fact that the target variables can have completely different ranges. Namely, if one of the target variables is in the range (0, 1) and another in the range (10, 100) and normalization is not used, then the values of the second variable will contribute much more to the overall score than the values of the first variable. In addition, weighting of the (normalized values of the) target variables so that the variance function gives more weight to some variables and less to others is supported. The prototype function (calculated at each leaf) returns as a prediction the tuple with the mean values of the target variables, calculated by using the training instances that belong to the given leaf.

3.3 Random Forests of Predictive Clustering Trees

An ensemble classifier is a set of base classifiers, which typically has a better performance than the individual classifiers. A new example is classified by combining the predictions of each classifier from the ensemble for that example. The predictions are typically combined by taking their average (for regression tasks) or their majority/probability vote (for classification tasks).

We use the random forests method to create the base classifiers in the ensembles. A random forest [18] is an ensemble of trees, obtained both by bootstrap sampling of the training set and by randomly changing the feature set during learning. More precisely, at each node in the decision tree, a random subset of

[3] The CLUS system is available for download at `http://clus.sourceforge.net/`.

the input attributes is taken, and the best feature is selected from this subset (instead of the set of all attributes). The number of attributes that are retained is given by a function f of the total number of input attributes x (e.g., $f(x) = x, f(x) = \sqrt{x}, f(x) = \lfloor \log_2 x \rfloor + 1$).

4 Codebook Construction Using Predictive Clustering

The architecture of the system for fast and scalable image retrieval using PCTs is presented in Figure 2. The systems consists of an off-line phase and an on-line phase. The off-line phase implements the construction of the visual codebook, the image descriptions and the search structure for the retrieval. The on-line phase implements the construction of the query image description, the querying of the images and presenting the result of the retrieval. In the remainder of this section, we discuss these phases in more detail.

The off-line phase starts with the generation of the local descriptors for the images. For each image in the database, affine-invariant Hessian regions are located [2]. Typically there are 3300 regions detected on an image of size 1024×768. For each of these affine regions, a 128-dimensional Scale invariant feature transform (SIFT) descriptor is then computed [12]. Next, the descriptors are used to create the visual codebook, which is the central part of the image retrieval system.

The proposed method for constructing the visual codebook is as follows. First, we randomly select a subset of the local (SIFT) descriptors from all of the images. Next, the selected local descriptors constitute the training set used to construct a PCT. For the construction of the PCT, we set the descriptive attributes (i.e., the 128 dimensional vector) to be also target and clustering attributes. Note that, this feature is a unique characteristic of the predictive clustering framework. To control the size of the visual codebook, we apply pre-pruning of the trees by requiring a given minimum number of instances/descriptors in each tree leaf. In order to get the desired number of leaves of a tree (i.e., visual words) for a given dataset, the number of required instances in a leaf can be easily estimated (roughly, it should be a bit smaller than the ratio between the number of training examples and the desired codebook size). Each leaf of the tree is a separate visual word and all of the leaves constitute the visual codebook. After the construction of the visual codebook, we sort all of the descriptors through the tree and count the number of descriptors that fall in a given leaf (i.e., correspond to a given visual word). We describe each image then with a histogram (sparse frequency vector) of the number of descriptors per visual word.

The PCTs are computationally efficient: it is very fast to construct them and to produce a prediction. However, the trees are unstable, i.e., the structure of the tree can change substantially for small changes in the training data [18]. To overcome this limitation and to further improve the discriminative power of the visual codebook, we use a small random forest that consists of four PCTs (similarly as in [8]). The final visual codebook is then obtained by concatenating the individual visual codebooks from each of the PCTs in the forest, thus the size of the final visual codebook is the sum of the sizes of the individual visual codebooks.

Once the visual codebook is constructed and the image descriptors are obtained, we proceed to create the search structure for the retrieval. For the search engine, we use the vector-space model of information retrieval [6]. The query and each image in the database is represented as a sparse vector of visual word occurrences. The search then calculates the similarity between the query vector and each image vector by using a L_2 distance. As a weighting scheme for the distance, we use the standard $tf - idf$ weighting scheme [6], which reduces the contribution to the relevance score of the words that occur commonly (since they are less discriminative).

For computational efficiency, the search engine stores the word occurrences in an tree-like index structure. The index structure maps the individual words to the images in which they occur. In the worst case, the computational complexity of querying the index structure depends linearly from the database size, but in practice it depends closely to linear from the number of images that match a given query. This presents a big saving of computational time. For sparse queries, this can result in even a more substantial speed-up, as only images which contain visual words present in the query need to be examined. The scores for each image are accumulated so that they are identical to explicitly computing the similarity.

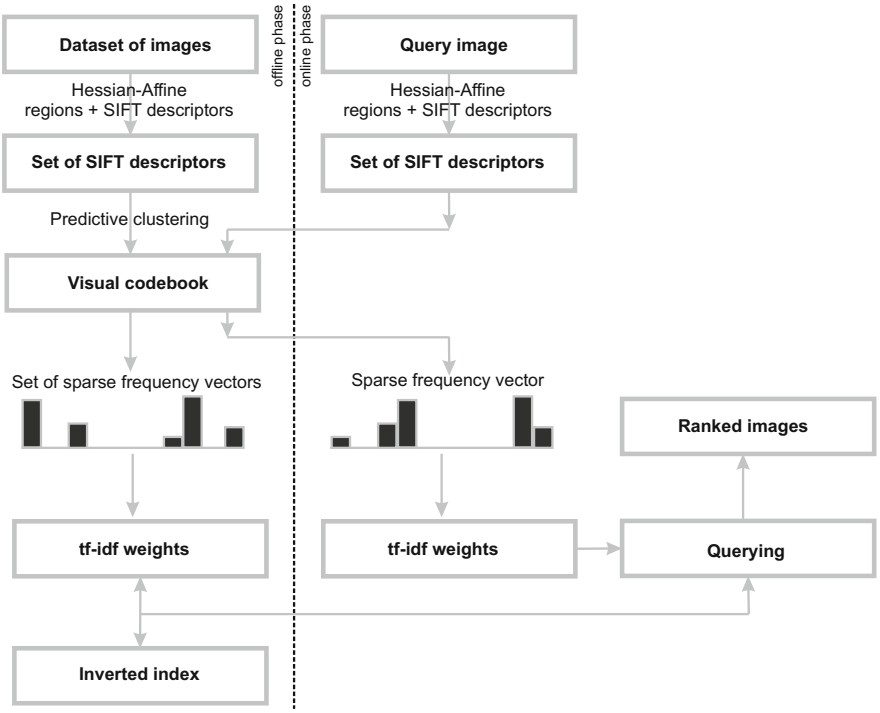

Fig. 2. Architecture of the proposed system for fast and scalable image retrieval based on predictive clustering trees

5 Experimental Setup

In this section, we present the experimental design we used to evaluate the proposed algorithm and compare it to other approaches. First, we present the dataset of images that we use. Next, we describe the evaluation metric we use to assess the retrieval performance. Finally, we state the experimental questions under investigation in this study.

5.1 Oxford Buildings Dataset

The Oxford Buildings dataset [19] is typically used as a benchmark dataset for large scale particular object retrieval. It consists of 5062 high-resolution images (1024×768) automatically downloaded from FLICKR by searching for 11 Oxford landmarks. The images are then manually annotated as to provide a solid ground truth for evaluation of the performance of retrieval systems. It also defines 55 queries (5 query objects/images for each of the 11 Oxford landmarks) that are used for performance evaluation. Each query consists of an image and query region/object of interest. This dataset is very challenging due to the substantial variations in scale, viewpoint and lighting conditions of the images and the objects that need to be retrieved.

In addition to this dataset often called Oxford $5K$, we use two other datasets to test the scaling abilities of the retrieval systems: Oxford $100K$ and Oxform $1M$ [2]. The Oxford $100K$ dataset contains 99782 high resolution images (1024×768), which were obtained from FLICKR by searching the 145 most popular tags. The Oxford $1M$ is created in a similar way, by crawling the 450 most popular tags from FLICKR and it consists of 1040801 images with medium resolution (500×333). These datasets are not challenging in terms of size, but also in the fact that they have much broader domain than the Oxford buildings: they include a mixture of different scenes, object, people, and buildings from all around the world.

5.2 Evaluation Measure

The most widely used performance measure for evaluation of methods in image retrieval is the *mean average precision* (mAP) [9,10]. We thus adopt mAP to evaluate the performance of our method for particular object retrieval and the discriminative power of its visual codebook.

The mAP is calculated as follows. For each of the 55 queries (5 for each of the 11 chosen Oxford landmarks), the average precision (AP) is computed as the area under the precision-recall curve ($AUPRC$) for that query. This score combines both precision and recall into a single performance score. Precision is defined as the ratio of retrieved positive images to the total number retrieved. Recall is defined as the ratio of the number of retrieved positive images to the total number of positive images in the dataset. An ideal precision-recall curve has precision 1 over all recall levels and this corresponds to an average precision of 1 and also $AUPRC$ of 1. The values of $AUPRC$ score are in the range [0,1]

and if they are bigger then the retrieval performance of the system is better. The overall mAP value is obtained by taking the mean of the average precisions of all queries and it is used as a single number to evaluate the overall performance.

5.3 Experimental Questions

We focus the experimental evaluation of the propose method on the following three research questions:

1. Does the use of predictive clustering trees for visual codebook construction improves the retrieval performance of the bag-of-visual-words approach compared to the widely used methods based on k-means and approximate k-means?
2. Whether the increase of number of descriptors and the visual codebook size influences the retrieval performance?
3. Does the proposed method for visual codebook construction is efficient and scalable to larger problems?

In order to answer these three question, we designed the following experimental setup. For answering the first question, we compare the performance of the visual codebook constructed using the random forest of PCTs with the performance of the one constructed using k-means and approximate k-means. This comparison is justified by the fact that most current results related to content-based particular object retrieval are obtained by using large visual codebooks created using k-means, while the current state-of-the-art results are obtained using approximate k-means.

We address the second question by constructing visual codebooks with different size and by using different numbers of local descriptors. More specifically, we use $800K$, $1M$, $5M$, $16.7M$ of local descriptors and produce codebooks with $10K$, $20K$, $50K$ and $1M$ visual words, respectively. The codebooks are constructed using our approach and the competing k-means and approximate k-means.

For answering the third question, we apply the visual codebook obtained by our method to the Oxford $100K$ and Oxford $1M$ image datasets. The performance results are compared with the results obtained using approximate k-means. We do not compared with exact k-means because it can't be scaled up to such a huge number of descriptors and visual words [2].

6 Results and Discussion

In this section, we present and discuss the results obtained from the experimental evaluation of the proposed method. First, we compare the performance of the proposed method to other methods from the literature. Next, we discuss the influence of the size of the codebook and the number of local descriptors considered to the retrieval performance of our system. Finally, we show the scalability of the proposed method by comparing its performance on a database with a million images.

Table 2 shows the results of the comparison of the three algorithms for visual codebook construction: k-means, approximate k-means (AKM) and random forest of predictive clustering trees (RF of PCTs). The results include experiments with a varying number of descriptors and a varying number of clusters. From the presented results, we can note that our method has a performance that is as least good as the one of the competing methods.

Table 2. Retrieval performance of k-means, approximate k-means and our algorithm based on predictive clustering trees (RF of PCTs) over the Oxford Buildings image dataset using different number of descriptors and visual words

Clustering parameters		mAP		
# of descriptors	codebook size	k-means	AKM	RF of PCTs
$800K$	$10K$	0.355	0.358	0.360
$1M$	$20K$	0.384	0.385	0.384
$5M$	$50K$	0.464	0.453	0.468
$16.7M$	$1M$	/	0.618	0.618

Furthermore, these results clearly show that the increase of the number of local descriptors and codebook size improve the retrieval performance of the systems. The best result (last row of Table 2) is obtained by using visual codebook with 1 million visual words obtained by all SIFT descriptors generated from all the images of the Oxford $5K$ dataset. To construct the codebook our method was running for $43.35h$, while the approximate k-means needed $69.9h$ (which is ~ 1.6 times slower). Considering that the proposed method based on random forests of PCTs is very computationally efficient [17], we can at small computational expense construct even larger visual codebook. An initial set of experiments indicates that further increase of the codebook size improves even more the retrieval performance.

Next, we evaluate the scalability of our method on the $5K$, $5K + 100K$ and $5K + 100K + 1M$ datasets using the $1M$ words visual codebook. The results are given in Table 3. Here, we compare our method to approximate k-means, since using the standard k-means is not feasible. From the results, we can see that the retrieval performance of our method is better than the one of approximate k-means. We can also note the drop in performance with the increase of the size of the image database. This is mainly due to the fact that these datasets now include much more noisy images outside of the domain of Oxford buildings.

We visually inspect the retrieval results and illustrate part of them in Figures 3 and 4. The first image in each row is the image that contains the query object, while the remaining four images are part of the retrieved images. The retrieval results given in Figure 3 reveals that the proposed method performs very well when there are considerable variations in the viewpoint, the image scale, the lighting and partial occlusion of the object.

On the other hand, the retrieval results given in Figure 4 illustrate examples in which the proposed systems fails to successfully retrieve the particular query

Table 3. Comparison of the discriminative power (with the mAP values) of the visual codebooks obtained by AKM and our method (RF of PCTs) over the three image datasets (both the algorithms use a codebook with $1M$ visual words)

Image dataset	AKM	RF of PCTs
$5K$	0.618	0.618
$5K$+$100K$	0.490	0.512
$5K$+$100K$+$1M$	0.393	0.401

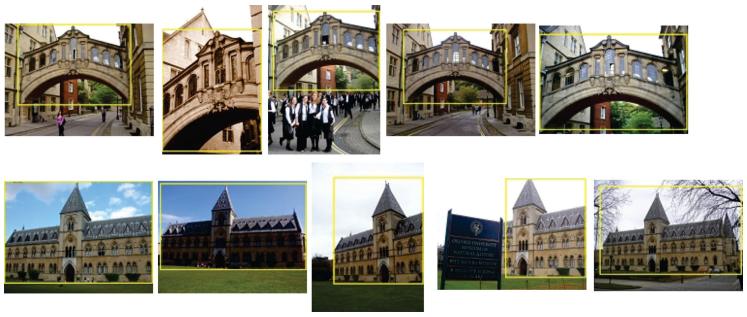

Fig. 3. Examples of searching the $5K$ dataset for: Bridge of sighs, Hertford College (first row), Pitt Rivers Museum (second row). First image in each row is the query image and the selected region with yellow color is the query object/building. The other four images in each row are result images obtained using the proposed system and algorithm.

Fig. 4. Examples of errors in retrieval for two query images (first images in the rows). The false positives for the first query is visually plausible, but the false positive for the second query is due to visual ambiguities in the local regions and the low numbers of visual words in the retrieved image.

object. For the first query (the first row of images), the failure is due to the presence of images that are visually plausible and consistent with the query image (the retrieved images also contain windows with bars).

The failure for the second query (the second row of images) is a result of the visual ambiguities present in the images (the first two result images) and the 'burstiness' effect (the second two result images) [20]. The visual ambiguities arise because the images have a very small number of local regions. This in turn results in a very large $tf - idf$ weights in the matching phase thus making an error while retrieving the particular object. The burstiness effect appears when a given local descriptors appear more frequently in an image than a model would predict. In this context, the burstiness distorts the image similarity measure (i.e., the $tf - idf$ weights) and thus pollutes the ranking of the images in the retrieval results. In our case, this is a result of the rich texture present in the resulting images such as water-drops and flowers.

7 Conclusion

In this paper, we present a method for fast and efficient construction of visual codebooks for particular object retrieval from large scale image databases. The construction of a codebook is an essential part of the bag-of-visual-words approach to image retrieval. It should thus be efficient and deliver a codebook with high discriminative power. However, the construction of a visual codebook is a bottleneck in the bag-of-visual-words approach, because it typically uses k-means clustering over millions image patches to obtain several tens of thousands visual words. Existing approaches are able to solve the efficiency issue, however, a part of the discriminative power of the codebook is sacrificed for better efficiency. In this paper, we propose to use predictive clustering trees (PCTs) for codebook construction. In this way, we efficiently construct visual codebooks and increase the discriminative power of the dictionary.

PCTs are a generalization of decision trees and are capable of performing predictive modelling and clustering simultaneously. More specifically, the method we propose uses PCTs for predicting multiple targets to construct the visual codebook – each leaf in the tree is a visual word. Furthermore, we construct a small random forest of PCTs to increase the stability of the codebook and its discriminative power. The overall codebook is obtained by concatenating the smaller codebooks from each tree.

We evaluated the proposed method on the Oxford buildings image database, which is a benchmark database for large scale particular object retrieval. We used three variants of the database that include $5K$, $100K$ and $1M$ images. We compare the proposed method to literature standard and state-of-the-art methods: k-means and approximate k-means clustering.

The results from the experimental evaluation reveal the following. To begin with, our method has a performance that is as least good as the competing methods on the smaller database with $5K$ images. Next, the increase of the number of local descriptors and codebook size improve the retrieval performance of the systems. Considering that the proposed method is computationally efficient, we can afford to construct larger codebooks that in turn will increase the retrieval performance. Finally, on the large databases, our method exhibits better retrieval performance than the competing approximate k-means. All in all, the

proposed method is able to efficiently produce a visual codebook with a high discriminative power.

We plan to extend this work along three major dimensions. First, we plan to extend the method and allow soft assignment of the descriptors to the visual words. Several studies have suggested that this could increase the retrieval performance of the system. Second, we will use the PCTs as search structures for performing the retrieval itself. This means that we will bypass the $tf - idf$ calculation and the creation of the inverted index, thus speed-up the retrieval even more. Finally, we will address the issue of burstiness by performing re-ranking of the top-ranked results by using spatial constraints. This procedure uses the predictions of the feature locations to estimate a transformation between the query region and each target image.

References

1. Liu, Y., Zhang, D., Lu, G., Ma, W.Y.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition 40(1), 262–282 (2007)
2. Philbin, J.: Scalable Object Retrieval in Very Large Image Collections. PhD thesis, University of Oxford, Oxford, UK (2010)
3. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2161–2168 (2006)
4. Jégou, H., Harzallah, H., Schmid, C.: A contextual dissimilarity measure for accurate and efficient image search. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
5. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: IEEE Conference on Computer Vision, pp. 1470–1477 (2003)
6. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press (1999)
7. Blockeel, H.: Top-down induction of first order logical decision trees. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1998)
8. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(9), 1632–1646 (2008)
9. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2010, VOC 2010 (2010), `http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html`
10. Nowak, S.: ImageCLEF@ICPR contest: Challenges, methodologies and results of the photo annotation task. In: International Conference on Pattern Recognition, pp. 489–492 (2010)
11. van de Sande, K., Gevers, T., Snoek, C.: Evaluating color fescriptors for object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(9), 1582–1596 (2010)
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
13. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: International Conference on Computer Vision, pp. 604–610 (2005)

14. Marée, R., Geurts, P., Wehenkel, L.: Content-based image retrieval by indexing random subwindows with randomized trees. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 611–620. Springer, Heidelberg (2007)
15. Uijlings, J., Smeulders, A., Scha, R.: Real-time bag of words, approximately. In: ACM International Conference on Image and Video Retrieval, pp. 1–8 (2009)
16. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
17. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition 46(3), 817–833 (2013)
18. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
19. The Oxford Buildings Dataset (2013), `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`
20. Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1169–1176 (2009)