Fast Codebook Generation by Sequential Data Analysis for Object Classification

Alexandra Teynor and Hans Burkhardt

Albert-Ludwigs-Universität Freiburg, Department of Computer Science
Georges Köhler Allee 052, 79110 Freiburg, Germany
{teynor,Hans.Burkhardt}@informatik.uni-freiburg.de

Abstract. In this work, we present a novel, fast clustering scheme for codebook generation from local features for object class recognition. It relies on a sequential data analysis and creates compact clusters with low variance. We compare our algorithm to other commonly used algorithms with respect to cluster statistics and classification performance. It turns out that our algorithm is the fastest for codebook generation, without loss in classification performance, when using the right matching scheme. In this context, we propose a well suited matching scheme for assigning data entries to cluster centers based on the sigmoid function.

1 Introduction

A lot of visual recognition systems use local features to identify members of visual object classes. They are characterized by their wide applicability and their robustness against variations in object appearance, shape and partial occlusions.

The locations for feature extraction are determined by different techniques, ranging from regular grids over interest point detectors to random locations and scales. Most commonly, interest point detectors are used, since they consider specific types of structures (e.g. blobs, edges, corners), and can also have a certain degree of invariance built in, e.g. scale or affine invariance [13]. Once regions of interest are found in images, different types of features can be extracted to describe these areas.

Typically, features obtained in this way are not used directly for learning, but they are clustered and so called "visual codebooks" are created. A huge variety of different approaches use visual codebooks at some step in the recognition chain, e.g. [2,4,12,14], just to mention a few.

There are mainly two reasons for the use of codebooks. One of them is to be able to deal with the huge number of local features extracted from the training images, especially if we want to compare the distribution of local appearance vectors in some feature space. For a typical multi class object recognition problem, hundreds of thousands of local representations might be extracted, which is too much to be handled directly by most algorithms. The other reason is to better model the variability of the different parts, by averaging over different examples. A single representation, typically the cluster mean, serves as representative for the different cluster members and can be used for training as well as classification.

G. Bebis et al. (Eds.): ISVC 2007, Part I, LNCS 4841, pp. 610–620, 2007.

For codebook generation, we would like to focus on dense regions in feature space, i.e. regions where common structures occur. A typical feature vector used for local description consists of 128 dimensions, e.g. SIFT [10] or GLOH [11]. Precise density estimation in this 128-dimensional space is prohibitive, since on the one hand, we would need an enormous mass of data, and on the other hand, the representation of this density would be very difficult. If we would choose, e.g., an unparametric representation in form of a histogram with each dimension quantized into 4 bins, we would end up with a $4^{128} = 1.2 \cdot 10^{77}$ dimensional feature vector, which is almost as much as the estimated number of atoms in the universe ($\sim 10^{80}$).

However, not all locations in this feature space are equally probable, so we only want to use those that are relevant for our problem. The goal of codebook creation in our context can be seen as to define a reduced partition of this high dimensional space. In this way, we are able to define relevant "parts" or "structures".

In this work, we propose a novel approach on how to obtain visual codebooks by identifying sufficiently dense regions in feature space given a certain similarity threshold and for creating clusters with low variance. The method is much faster than other commonly used clustering algorithms as K-means or agglomerative clustering and can therefore be used to process more local features.

The outline of this paper is as follows: first, we give an overview about related work in section 2, then we describe our approach in section 3. In section 4, we show some experiments which are discussed subsequently. Finally, the conclusions are drawn in section 5.

2 Related Work

A variety of different clustering algorithms have been applied to visual codebook creation, e.g. hierarchical clustering (divisive clustering [9] as well as agglomerative clustering), clustering based on function optimization (e.g. expectation maximization (EM) type clustering [15]) or mixed techniques [8]. A general overview about clustering algorithms can be found in [15]. The two most commonly used techniques for codebook generation are agglomerative hierarchical clustering as well as K-means clustering. We review them briefly here. The task is always to cluster a set of local features $\mathbf{x}_n, n = 1, \ldots, N$.

1. **Agglomerative clustering**
   In the beginning, all data entries are regarded as single clusters. In each subsequent step, the most similar clusters are grouped, until only a single cluster remains. In this way, a tree structure of the data is created. To obtain individual clusters, the tree is "cut", either according to the desired number of clusters or a given minimum similarity between cluster members. In order to determine the similarity between clusters, different linkage strategies can be applied. Typically, the average link paradigm is used as it produces compact clusters, although it has a rather high time ($O(N^2 \log(N))$) and space

complexity ($O(N^2)$). This method is, for example, applied by Agarwal et al. [1] and Leibe et al. [7].

2. **K-means clustering**
   K-means clustering is an iterative procedure, where a function J describing the within-cluster variance gets minimized:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N_k} d(\mathbf{x}_{ij}, \boldsymbol{\mu}_j) \tag{1}$$

   We have K clusters, each consisting of N_k members, $\boldsymbol{\mu}_j$ is the cluster mean and $d(\cdot, \cdot)$ is a distance function. The time complexity of this algorithm is $O(NKq)$, where q is the number of iterations needed. The main advantage of K-means is its simplicity, however, it is sensitive to outliers. The number of clusters has to be fixed a priori, and the cluster means might lie far away from the cluster members. When random initialization is used, the clustering result might differ between runs. K-means clustering is used, for example, by Weber et al. [16] and Lazebnik et. al [6].

3 Proposed Approach

3.1 Sequential Clustering

Our goal is to find a partitioning of a high dimensional feature space for part based object class recognition. Typical clustering algorithms as described in the previous section do in fact more than that. They try to recover the structure of the data in feature space, e.g. by building a tree or minimizing an error criterion. For common objective functions, as in equation 1, this results in a higher number of cluster centers in more densely populated regions in feature space.

If we follow the principle of Occam's razor, we should select the simplest method that solves our problem. We only need to identify "sufficiently dense" regions in feature space and distribute cluster centers in these areas. We propose a simple sequential algorithm with low runtime complexity. The basic idea is to create hyperspheres with a certain radius. As all clustering algorithms, we assume that the distance in feature space does resemble the visual similarity of the patches. So the radius to be chosen depends on the distance in which samples are still considered visually similar. This has to be done experimentally.

The proposed algorithm is based on the Modified Basic Sequential Algorithmic Scheme (MBSAS) described in [15]. It is a two pass algorithm where first candidate cluster centers are determined. Then, the data is assigned to the respective closest cluster centers. After all data has been assigned, new cluster representatives are calculated from the cluster members in order to represent them better. Clusters with too few members get discarded. The algorithmic description can be found in algorithm 1. There, C denotes a set of features and $|C|$ the cardinality of the set C.

The difference to the original algorithm described in [15] is that the cluster centers are calculated after the assignment of all members, and only if the minimum member constraint has been fulfilled. This further speeds up computation.

Algorithm 1. Modified Sequential Clustering for Codebook Generation

Input: patch features $\mathbf{x}_n$, $n = 1, \ldots, N$; hypersphere radius ϵ; min density θ ;
Output: codebook entries $\mathbf{k}_l$, $l = 1, \ldots, K$
begin
  $m \longleftarrow 1$;
  $C_m \longleftarrow \{\mathbf{x}_1\}$;
  for $i = 2$ **to** N **do**
    Find $C_k : d(\mathbf{x}_i, C_k) = min_{1 \leq j \leq m} d(\mathbf{x}_i, C_j)$;
    if $d(\mathbf{x}_i, C_k) > \epsilon$ **then**
      $m \longleftarrow m + 1$;
      $C_m \longleftarrow \{\mathbf{x}_i\}$;
    end
  end
  for $i = 1$ **to** N **do**
    Find $C_k : d(\mathbf{x}_i, C_k) = min_{1 \leq j \leq m} d(\mathbf{x}_i, C_j)$;
    $C_k \longleftarrow C_k \cup \{\mathbf{x}_i\}$;
  end
  $l \longleftarrow 1$;
  for $i = 1$ **to** m **do**
    if $|C_i| \geq \theta$ **then**
      $\mathbf{k}_l \longleftarrow$ cluster representative for C_i;
      $l \longleftarrow l + 1$;
    end
  end
  $K \longleftarrow l - 1$;
end

The result of the clustering algorithm will depend on the order of the input. To not bias the result, the features should not be fed to the algorithm in the order they were extracted from the images, but shuffled beforehand. In order to determine the cluster representative, different methods are possible. In this work, the mean vector of the cluster members is taken, but also the median could have been used. The time complexity of MBSAS is $O(NK)$, which is smaller than the comlexity of agglomerative or K-means clustering. Please note that for calculating the time complexity, K is the initial number of candidate clusters generated in the first part of the algorithm, not the final number of valid clusters. How significant the speed up is can be seen from the actual clustering times for sample datasets in section 4.3.

3.2 Matching to Codebook Entries

In order to assign newly extracted features to codebook entries, different methods can be applied. We have tested two commonly used approaches, namely nearest neighbor and threshold based matching. We also applied a weighted matching scheme based on the sigmoid function, which we found very suitable.

- **N nearest neighbor matching:**
  The feature vector gets matched to the n nearest cluster centers, no matter what the distance is. We use $n = 3$ in our experiments.

- **Threshold based matching:** The features match to all cluster centers that are within a certain threshold. The threshold used is the hypersphere radius used for clustering. Thus, a vector might match to zero, one or more clusters.
- **Weighted matching using a sigmoid function:** A new feature vector $\mathbf{x}$ matches to a codebook entry k_l with the weight w_l determined by a sigmoid function:

$$w_l = \frac{1}{1 + e^{\alpha(d(\mathbf{k}_l, \mathbf{x}) - \epsilon)}}$$

  The rationale behind this assignment function is that within a certain radius ϵ, the patches are all visually similar and should get the same high matching score. Patches with a distance above a threshold are too dissimilar and should get a low matching score. The region in between can be modelled by the factor α. It determines how steep the sigmoid function is.

For histogram creation, the matching values get normalized to sum up to one, i.e. each feature contributes the same to the distribution.

4 Experiments

In order to show the performance of our approach, we conduct different experiments. We use two different databases, each with a different classification task. One dataset is the Caltech3[1] dataset (airplanes, faces, motorbikes), the other is the Caltech101 dataset[2].

For each image in the respective database, we extract Harris-Laplace and Hesse-Laplace interest points [13]. The Harris-Laplace detector fires on corners, where the Hesse-Laplace detector finds blob like structures. The two types of interest points are treated separately in order to verify that the qualitative results do not depend on the type of interest point detector used. For each region, we calculate rotation sensitive GLOH [11] descriptors. For the computations, the binaries provided by Mikolajczyk are used[3].

The similarity threshold for the MBSAS clustering and the hard histogram matching was determined as follows: pairs of random sample patches from the database were shown to 7 different individuals who had to judge the visual similarity of the patches being "very well", "well", "not sure", "dissimilar" or "very dissimilar". The threshold was set between the average Euclidean distance in feature space for pairs that were judged to match "very well" and "well".

From each database, 30000 random patches were drawn from the training images and clustered with the MBSAS, K-means and agglomerative clustering scheme. This number was mainly limited by the time complexity of the agglomerative clustering algorithm. For the Caltech101 database, we used 15 randomly selected training and test images. We drew three independent sets and averaged

[1] from http://www.robots.ox.ac.uk/~vgg/data3.html
[2] from http://www.vision.caltech.edu/Image_Datasets/Caltech101
[3] from http://www.robots.ox.ac.uk/~vgg/research/affine/

the respective results. For the Caltech3 database, the same training and test images were used as in [5].

We discard single member clusters for all clustering results, since they are considered as too uncommon to generalize well. So $\theta = 2$ in our case. In order to have comparable codebook sizes, the cut value for the agglomerative clustering was chosen such that after the removal of the single member clusters the cluster number is the same as for the MBSAS clustering. The initial number of clusters for the K-means clustering was set so that the resulting number of non single member clusters was as close as possible to the value of the two other approaches. Since our K-means algorithm uses random initialization, it is hard to obtain exactly the same number.

4.1 Codebook Statistics

First, we want to look at certain statistics of the codebooks generated. We list the number of clusters obtained after the removal of the single member clusters as well as the single cluster ratio (scr), i.e. the percentage of the clusters that contained just a single member. For each cluster, we compute the cluster variance, i.e. the average squared distance of the cluster members to the cluster center. We list the average cluster variance per codebook and also the distribution of the cluster variances. The results can be seen from table 1 for the Caltech3 database, and from table 2 for the Caltech101 database.

The results are very consistent across the different databases and interest point detector types. The clustering with MBSAS results in visually very compact clusters, with a low average cluster variance. As can be seen from the variance distribution, there are no clusters with a very big variance, as e.g. for the K-means clusters. Cluster centers obtained as an average from widely spread data points are not guaranteed to represent the members adequately. This can also be confirmed by visual inspection of the clusters: patches belonging to some K-means clusters are visually quite distinct. As a consequence, the single cluster ratio is quite high for MBSAS codebooks as opposed to the other approaches, since only areas with a certain part density in a small neighborhood are kept.

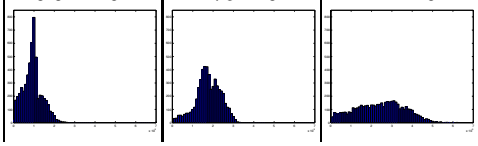
4.2 Classification Results

In order to compare the codebooks from a qualitative point of view, we performed two classification tasks. We first solve a two class problem on the Caltech3 database, where objects have to be distinguished from a background class. We then deal with a multi class problem on the Caltech101 database.

Since we only want to test the quality of the codebooks obtained, we use a simple "bag of feature" approach: we create histograms of object parts using the different codebooks and matching strategies. We neglect any spatial information. For classification, we use a standard SVM implementation (libSVMTL[4]) with a histogram intersection kernel. For the multi class problem, a one-vs-rest SVM was used.

[4] http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/

Table 1. Cluster statistics for codebooks for the Caltech3 database: scr = single cluster ratio; avg var = average variance; var dist = variance distribution

Hesse-Laplace	MBSAS	agg	K-means
# clusters	5489	5489	5005
scr	0.71	0.40	0.33
avg var	$0.94 \cdot 10^6$	$1.76 \cdot 10^6$	$2.42 \cdot 10^6$
var dist			

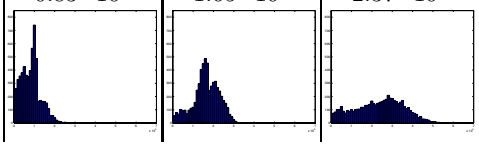
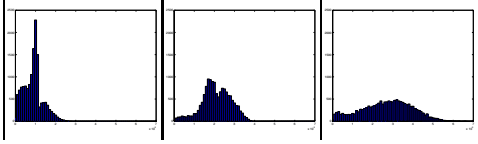
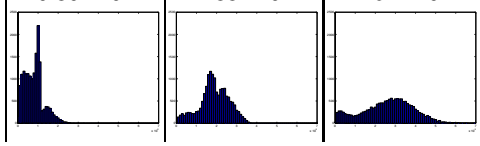
Harris-Laplace	MBSAS	agg	K-means
# clusters	5817	5817	5540
scr	0.69	0.39	0.26
avg var	$0.83 \cdot 10^6$	$1.65 \cdot 10^6$	$2.37 \cdot 10^6$
var dist			

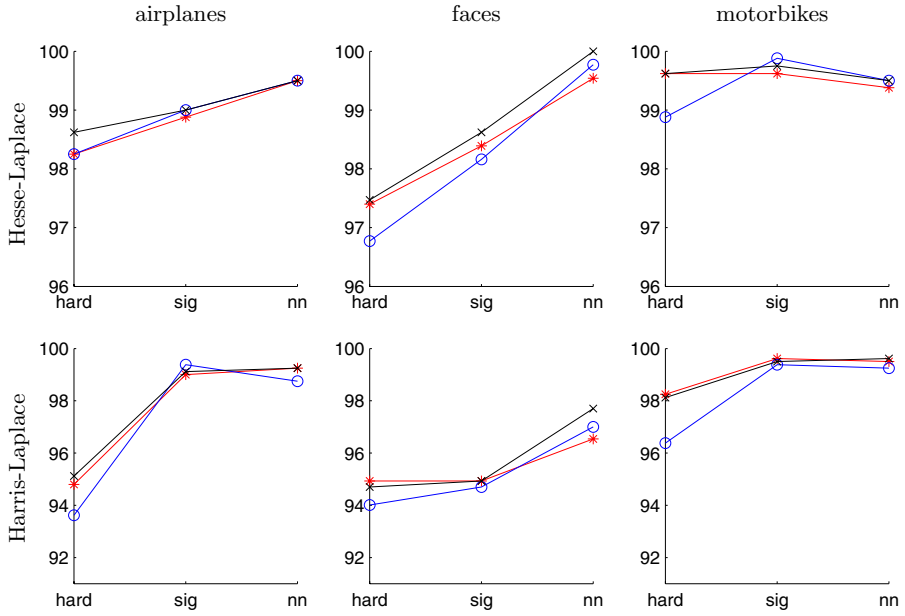
Table 2. Cluster statistics for codebooks for the Caltech101 database: scr = single cluster ratio; avg var = average variance; var dist = variance distribution

Hesse-Laplace	MBSAS	agg	K-means
# clusters	4917	4917	4967
scr	0.77	0.38	0.34
avg var	$0.89 \cdot 10^6$	$2.09 \cdot 10^6$	$2.71 \cdot 10^6$
var dist			

Harris-Laplace	MBSAS	agg	K-means
# clusters	5490	5490	5518
scr	0.75	0.38	0.26
avg var	$0.80 \cdot 10^6$	$1.88 \cdot 10^6$	$2.64 \cdot 10^6$
var dist			

The two class problem on Caltech3 is relatively easy as the images contain distinct structures for the individual object classes. Caltech101 is more diverse and contains a variety of different structures. The classification results for the different interest point detector types and matching strategies can be seen from table 3 for the Caltech3 database and from table 4 for the Caltech101 database.

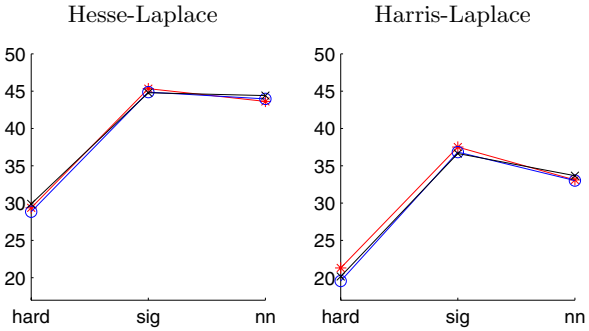
Table 3. Classification rate in % for the different Caltech3 problems. Results for MBSAS codebooks are shown in blue(o), for agglomerative codebooks in red (*) and for K-means codebooks in black(x). The results are given for the different matching strategies: hard = hard, sig = sigmoid and nn = 3 nearest neighbor.



The overall classification performance for Caltech3 is very well, in particular we could obtain a classification rate of 100% for the faces class with Hesse-Laplace interest points and K-means clustering. In general, Hesse-Laplace interest points performed better than the Harris-Laplace interest points. For Caltech101, more sophisticated classification strategies incorporating also spatial information brought better results (see e.g. [6]). However, in this work we only want to compare the relative performance of different clustering schemes.

From a classification point of view, there is no real winner in the clustering scheme. For the Caltech3 database and sigmoid matching, the MBSAS codebooks are slightly superior compared to the others regarding the categories airplanes and motorbikes, for the faces category, K-means clustering is superior when using nearest neighbor matching. Agglomerative clustering gives best results for the Caltech101 database and sigmoid matching.

Table 4. Classification rate in % for the Caltech101 problem. Results for MBSAS codebooks are shown in blue(o), for agglomerative codebooks in red (*) and for K-means codebooks in black(x). The results are given for the different matching strategies: hard = hard matching, sig = sigmoid matching and nn = 3 nearest neighbor matching.



Using hard matching with a tight threshold typically gives inferior results compared to using sigmoid or nearest neighbor matching. Here the MBSAS clusters are especially sensitive. When the structures are distinct as in the Caltech3 database, nearest neighbor matching is superior in almost all cases, since the nearest parts matched are likely to be from the same class. For more diverse databases as the Caltech101 sigmoid matching performed best.

4.3 Run Times

In this section, we list experimental run times for the different approaches. We performed the clustering for a different number of GLOH features computed around Hesse-Laplace interest points extracted from the Caltech101 dataset. The processing times were measured on 2.6 GHz AMD opteron processors. The results are listed in table 5. For the agglomerative and K-means clustering, we used the C clustering library by de Hoon et al. [3]. In this implementation, the K-means algorithm iterates until the assignment of features to clusters does not change any more. The MBSAS implementation was done by ourselves. The run times for the agglomerative clustering represent the time the entire tree needs for building, the run times for K-means and MBSAS clustering are given for settings that result in about the same number of clusters. For larger amounts of data, we increased the required number of member for clusters to be valid. We can observe that the processing time for K-means and MBSAS clustering grows over-proportional to the number of features. This is due to the fact that for the K-means algorithm, the number of iterations (q) until convergence is larger, and for the MBSAS algorithm, the number of candidate clusters in the first part of the algorithm is larger. The MBSAS algorithm runs very fast compared to the other algorithms. Thus it is possible to use more local representations in order to get a more complete view on the data. When increasing the hypershpere radius ϵ in which structures are considered similar, an even larger speed up is possible.

Table 5. Experimental run time results for different numbers of local features and clustering schemes

# local features	10^4	$3 \cdot 10^4$	10^5	$5 \cdot 10^5$
# of clusters	~ 750 ($\theta = 2$)	~ 3000 ($\theta = 2$)	~ 3700 ($\theta = 4$)	~ 5000 ($\theta = 10$)
Agg. clustering	26.6 min	11.6 h	n.a.	n.a.
K-means clustering	15.3 min	4.0 h	36.3 h	n.a.
MBSAS clustering	1.9 min	16.7 min	2.0 h	45.6 h

5 Conclusions

In this work we have presented a novel scheme to obtain codebooks for part based object classification. We compared our method to other commonly used algorithms for codebook creation. Our experiments have shown that despite the different properties of the resulting clusters, all three approaches performed similarly in a bag of feature classification approach. It seems to be sufficient to have cluster centers distributed in about the right area of feature space. Following the principle of Occam's razor, we have shown that no complicated algorithms with huge memory and runtime requirements are necessary, a simple sequential clustering scheme is sufficient. So more local structures can be used in codebook generation, to get a more complete view on the data distribution.

We have also shown that the matching scheme has more influence on recognition performance than the clustering algorithm: for diverse structures, sigmoid matching has shown to be superior, but also simple nearest neighbor matching is well suited, especially for simple problems.

All these observations were made for a bag-of-feature type classification approach. We plan to also test whether these observations hold for geometry based approaches, where distinct object parts have to be selected from codebooks.

Acknowledgments

This work was partially funded by the European Union MUSCLE NoE (FP6-507752).

References

1. Agarwal, S., Awan, A., Roth, D.: Learning to Detect Objects in Images Via a Sparse, Part-Based Representation. IEEE TPAMI 26(11), 1475–1490 (2004)
2. Agarwal, S., Roth, D.: Learning a Sparse Representation for Object Detection. In: Proc. ECCV (2002)
3. de Hoon, M.J.L., Imoto, S., Nolan, J., Miyano, S.: Open Source Clustering Software. Bioinformatics 20(9), 1453–1454 (2004)
4. Li Fei-Fei, R., Fergus, R., Perona, P.: One-shot Learning of Object Categories. IEEE TPAMI 28(4), 594–611 (2006)

5. Fergus, R., Perona, P., Zisserman, A.: Object Class Recognition by Unsupervised Scale-Invariant Learning. In: Proc. CVPR, vol. 2, pp. 227–264 (2003)
6. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: Proc. CVPR, vol. 2, pp. 2169–2178 (2006)
7. Leibe, B., Leonardis, A., Schiele, B.: Combined Object Categorization and Segmentation with an Implicit Shape Model. In: Proc. of the Workshop on Statistical Learning in Computer Vision, Prague, Czech Republic (May 2004)
8. Leibe, B., Mikolajczyk, K., Schiele, B.: Efficient Clustering and Matching for Object Class Recognition. In: Proc. BMVC (2006)
9. Linde, Y., Buzo, A., Gray, R.: An Algorithm for Vector Quantizer Design. Communications, IEEE Transactions on 28(1), 84–95 (1980)
10. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV 60, 91–110 (2004)
11. Mikolajczyk, K., Leibe, B., Schiele, B.: Local features for Object Class Recognition. In: Proc. ICCV, vol. 2, pp. 1792–1799 (2005)
12. Mikolajczyk, K., Leibe, B., Schiele, B.: Multiple Object Class Detection with a Generative Model. In: Proc. CVPR (2006)
13. Mikolajczyk, K., Schmid, C.: Scale and Affine Invariant Interest Point Detectors. IJCV 60(1), 63–86 (2004)
14. Opelt, A., Pinz, A., Zisserman, A.: A Boundary-Fragment-Model for Object Detection. In: Proc. ECCV (2006)
15. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 3rd edn. Academic Press, London (2006)
16. Weber, M., Welling, M., Perona, P.: Unsupervised Learning of Models for Recognition. In: Proc. ECCV, Dublin, Ireland (2000)