Vocabulary Tree Hypotheses and Co-Occurrences

Martin Winter, Sandra Ober, Clemens Arth, and Horst Bischof

Institute for Computer Graphics and Vision, Graz University of Technology
{winter,ober,arth,bischof}@icg.tu-graz.ac.at

Abstract

This paper introduces an efficient method to substantially increase the recognition performance of a vocabulary tree based recognition system. We propose to combine the hypothesis obtained by a standard inverse object voting algorithm with reliable descriptor co-occurrences. The algorithm operates on different depths of a standard k-means tree, coevally benefiting from the advantages of different levels of information abstraction. The visual vocabulary tree shows good results when a large number of distinctive descriptors form a large visual vocabulary. Co-occurrences perform well even on a coarse object representation with a few number of visual words. We demonstrate the achieved performance increase, robustness to occlusions and background clutter in a challenging object recognition task on a subset of the Amsterdam Library of Object Images (ALOI).

1 Introduction

Besides a number of well known (and partially solved) problems in computer vision (*e.g.* illumination conditions, occlusions, viewpoint changes *a.s.o.*), recent research interest has focused on the problem of object recognition in large databases. Encouraging recognition rates have been achieved using tree-like representations of local descriptors.

One typical example is the approach of David Lowe [12], who organized SIFT descriptors from all training images in a kd-tree with a best-bin-first modification to find approximate nearest neighbors to the descriptor vectors of the query. The correspondences of the matched descriptor pairs of the query and kd-tree patches have to be confirmed or rejected in further verification and consistency checks. Another approach is the one from Lepetit *et.al.* [11], who were the first to introduce multiple randomized trees as classification technique. Their approach mainly focused on the detection of a single object, which is fast enough for real-time applications. Obdrzalek and Matas [18] used a binary decision tree to index keypoints and minimize the average time to decision. The leaves of the tree represent a few local image areas where every inner node is associated with a *weak classifier*. They demonstrated the robustness of their method to background clutter, occlusion, and large changes of viewpoints with hundreds of objects. Nistér and Stewénius [17] presented an approach where hundred thousands of local descriptor vectors are quantized in a hierarchical vocabulary tree. It is able to organize a database of 1 million images. They presented a scoring scheme which results have to be verified in an additional post-verification step using the geometry of the matched keypoints of the n top ranked objects to improve the retrieval quality.

Common to most of the approaches mentioned is the need for such a post-verification algorithm to guarantee for an acceptable performance rate and stable recognition results. A popular method is using geometrical constraints for eliminating false positives and strengthening correct hypotheses [17]. Another possibility are consistency checks of local neighbor relations of interest points [22].

While all of these algorithms require additional computational overhead, the information we incorporate into our system can be obtained almost for free from our own tree-based representation. In particular, we build a hierarchical vocabulary tree and apply inverse voting similar to Nistér and Stewénius [17]. The inverse voting uses the leaves of the tree. Another very coarse representation is taken from a lower tree level. To obtain a high distinctiveness of that representation, we use a very efficient, yet memory and computationally efficient specificity of spatial relations, namely co-occurrences of descriptors. A rather simple, but effective arbitration strategy is used to foster the hypotheses of inverse voting.

In the following section we describe the building process for our vocabulary tree and section 3 outlines the co-occurrence representation. The final decision rules are described in section 4. Finally, we demonstrate the power of our approach in the experimental section 5.

2 Building the Vocabulary Tree

2.1 Feature detection and description

In our approach (depicted in Fig. 1), we use Lowe's *Difference of Gaussian* (DoG) detector together with SIFT-keys [12] to obtain rather accurate keypoints with high repeatability [7]. DoG keypoints have been proposed together with SIFT-keys and show excellent recall performance [13, 15].

The DoG-detector takes the differences of Gaussian blurred images as an approximation of the scale normalized Laplacian and uses the local maxima of the responses in scale space as an indicator for a keypoint. We mention, that the approach is not restricted to DoG-points. Any other keypoint detector with high repeatability can be used instead. SIFT-descriptors are quantized gradient histograms. Those gradient histograms are calculated in a subdivided

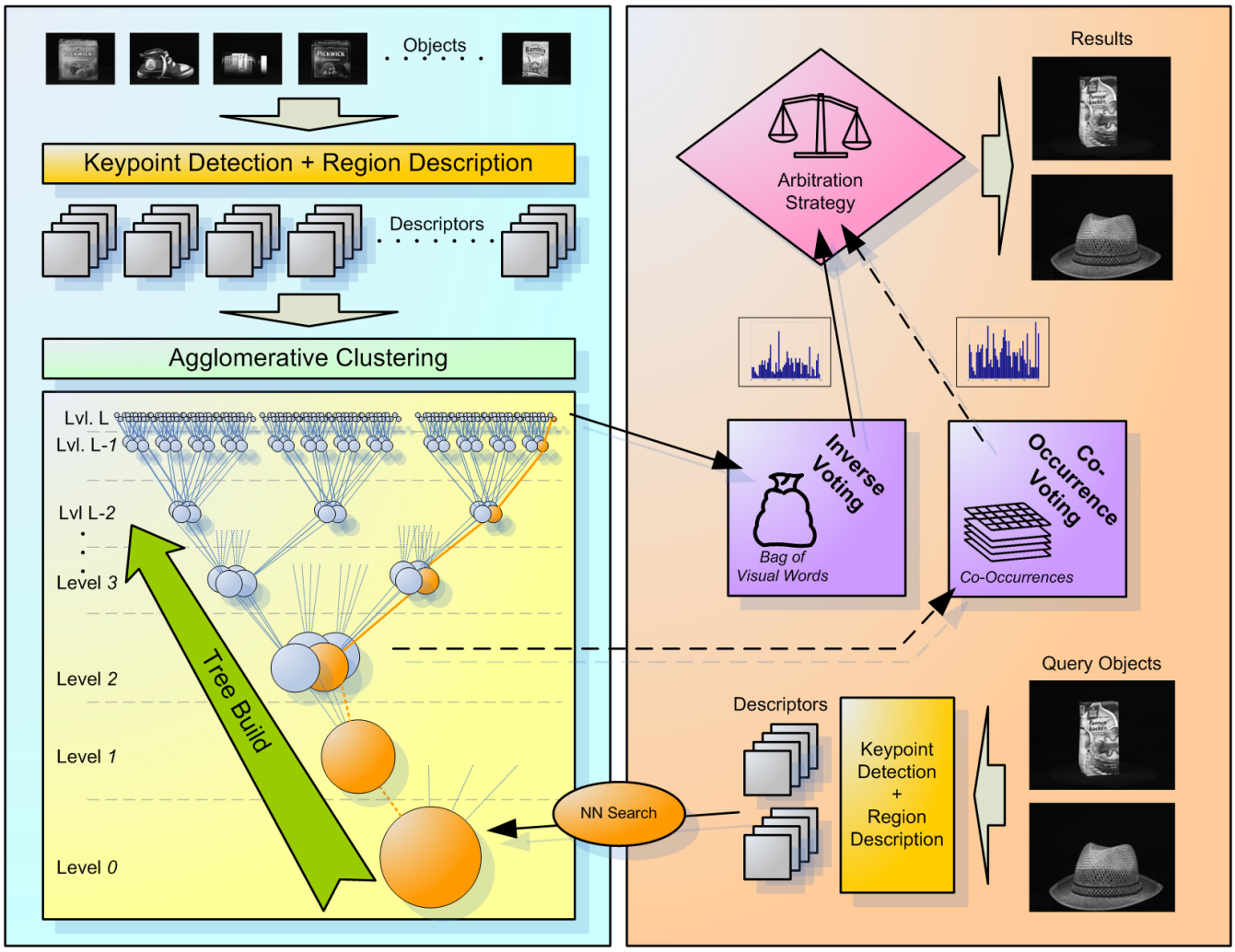

Figure 1: Diagram of our object recognition framework. On the left, the training process is illustrated. For generating the visual vocabulary, a limited number of objects is chosen and descriptor extraction is followed by agglomerative clustering and building a k-means tree ($k = 4$ in the case depicted here). The tree is drawn upside-down to make clear that the outcome of the agglomerative clustering algorithm forms the leaves of our k-means tree. For all objects in our database, we extract descriptors, store the nearest neighbor matches in an *inverted file structure* and coevally extract co-occurrence information from a lower level of our tree (level two in the case described here). Querying works as depicted on the right side of our diagram. Descriptors are extracted from our query object and the nearest neighbors in our tree are used to build an inverse voting result, simultaneously collecting co-occurrences for our second voting algorithm. A final arbitration strategy is used to draw the final decision.

patch in order to cover spatial information. The descriptor dimensionality is 128. In order to be robust against pixel noise and to avoid the detection of too large regions, we restricted the scale of the obtained DoG-points.

2.2 Building the vocabulary tree

We use a hierarchical k-means tree as data structure for fast indexing and retrieval of descriptors as illustrated in Figure 1. Instead of building the tree with hundred thousands of descriptors, we propose a tree, where a lower number of visual words act as leaves, because Nistér and Stewénius have shown in [17], that for more than 100K leaf nodes no substantial performance increase can be expected. Therefore we quantize our feature descriptor vectors with unsupervised *agglomerative hierarchical clustering* to obtain the visual words which will act as leaf nodes in our vocabulary tree and used the proposed *Average-Link algorithm with RNNs* of Leibe *et.al.* described in [10], which has feasible runtime properties and can deal which such a large number of descriptors. The main advantage of this algorithm is the fact, that we have to select only one parameter to tune the quantization properties as we can easily select the tolerated dissimilarity of two points belonging to the same cluster in the descriptor space.

These visual words are quantized in a repeated k-means clustering with a fixed k down the levels of the hierarchical tree. At every node the set of descriptor vectors clustered by k-means is partitioned in k nodes and propagated to the next level until no further splitting is possible. The number of visual words, that can be represented is k^L, where k is the branch factor and L is the deepest level of the tree as illustrated in Figure 1.

Pre-clustering of descriptors into visual words decreases the computational overhead for building the hierarchical tree. Furthermore, we get a very well balanced tree which will allow us to index and retrieve images or objects we never had in the training set of the descriptor vectors for building the tree. In our experiments we show the power of this vocabulary tree which has to be built only once. Thus we can reduce the time spent for building the tree (including agglomerative pre-clustering) from several days to a few hours.

2.3 Implementation of indexing

We have to take notice that we want to build a very balanced tree where the training and recognition of new objects is possible without any retraining of the tree. New objects are learned and inserted in the tree by firstly detecting feature descriptor vectors. Secondly, every descriptor vector starts at the root level of the vocabulary tree and will be compared with the next possible node clusters of the next level l using L_2-norm. After the next nearest cluster node is selected the feature descriptor goes down the tree level per level until it will end at a leaf node.

Because every leaf node has a unique index, we can represent every image as a set of index numbers like a fingerprint. Those index numbers are used to store all those pre-matches in an *inverted file structure* (IFS), which is a very efficient way to handle recognition in large image databases. Every index of this IFS is assigned with all object- or image ID-number where their descriptor vectors have matched with this leaf node. The vocabulary tree and the IFS will be needed for the later recognition step.

We can clearly see that insertion of further objects is very easy and fast. The tree is usually not very deep and the number of node clusters k is chosen to be very small (typically 8 to 12). So the number of comparisons by traversing the descriptor vector through the tree is obviously very low and therefore we can achieve very high performance rates not only during the training (indexing) but also during the recognition (scoring) step.

2.4 Generation of vocabulary tree hypotheses

To recognize an object or image with the vocabulary tree it is obvious, that we can use the same routine as for indexing. We detect feature descriptor vectors and propagate them down the vocabulary tree by comparing the descriptor vector to the k nearest node centers and choosing the nearest one for the next level. As explained before we collect the number of leaf indices for our unknown object or image similarly to indexing. We use those gathered indices with our *inverted file structure* to set up a scoring table for each object or image. This scoring table gives us an object voting list ranked by the number of found matched feature descriptor vectors. This first hypothesis has to be normalized by the number of feature descriptor vectors for each object or image used in the indexing step. This is very important, because we can achieve fairness for every database object or image to be recognized even if the number of descriptor vectors is very low and therefore the number of occurrences in the IFS is very sparse. We also investigated other methods of scoring, but we could not achieve substantially better results. This very fast generated hypothesis has to be verified by another step to improve the retrieval quality. Instead of using a separated computationally demanding method like using the geometry of the matched keypoints or counting the number of matching nearest neighbors, we present a more efficient way to verify generated hypotheses by boosting them with co-occurrences gathered online at a lower tree level.

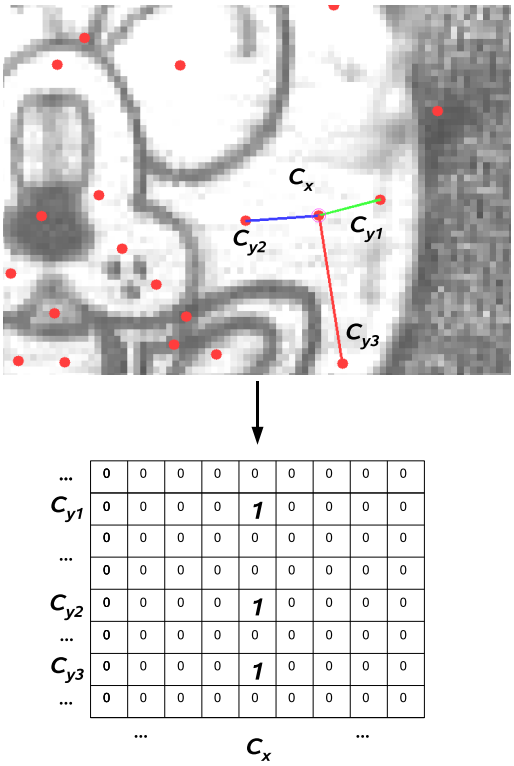

Figure 2: Illustration of finding the nearest neighbors in image space and corresponding entries to the co-occurrence matrix for a certain keypoint. The centers of the DoG-points are indicated by red dots.

3 Co-Occurrences of descriptors

Spatial relations among keypoints have already been investigated by many authors (e.g. [1, 2, 3, 4, 5, 6, 9, 14, 16, 19, 20, 21]) and it has been shown, that they can significantly improve recognition performance. In our approach, we use a very simple specificity of spatial relations which can be efficiently obtained from a coarse level of the already calculated k-means tree (see Figure 1 for illustration). Moreover, we use a very extreme form of co-occurrences as we represent only the presence or absence of co-occurrences. The intuition behind is, that the co-occurrence of descriptors is very discriminative because it is very unlikely, that two descriptors co-occur just by chance.

In particular, we take the best matching cluster center in a very low depth k-means tree level already calculated(!) so that for every keypoint in the image a corresponding cluster index is stored. There is nearly no additional computational effort necessary to extract that representation out of the tree. Please note, that the number of cluster indices is substantial

lower than the number of clusters used for the inverse voting approach. While building the vocabulary tree in an off-line calculation, we obtain n cluster indices in a depth l of the tree (k is the branch factor indicating the number of children for every node):

$$n = k^l \tag{1}$$

In our experiments, we have chosen a branch factor of $k = 9$ to obtain best results and selected tree levels $l_c = 2, 3, 4$ for co-occurrences thus causing $n_c = 81, 729$ or 6561 co-occurrence cluster indices. This is about a factor of 20, 190 or 1600 less cluster indices than visual words used for the inverse voting.

To calculate the co-occurrence matrix, we simply identify the nearest neighbor for every keypoint in image space. Thus, every co-occurrence is identified by a pair of cluster indices which we insert into the two-dimensional co-occurrence matrix. Figure 2 illustrates this procedure. The nearest neighbor property of certain interest points in the image space is sometimes violated by spurious highlights, unstable detection of keypoints and of course aspect changes introduced by different viewpoints. We alleviate this problem by entering the n nearest neighbors (typically $n = 3$) in the co-occurrence matrix.

For a reasonable amount of cluster centers, the co-occurrence matrix is sparsely populated and typically only a few permils of all the possible co-occurrences are assigned. Multiple occurrences are even much more unlikely so that it is possible to limit the entries to the binary information, whether a specific co-occurrence is observed for a certain object or not. This binary coding and a sparse storage scheme allow us to reduce the necessary amount of memory to a minimum.

To build the full representation for a single object (multiple viewpoints), all the co-occurrences of the trained images are entered in one single two-dimensional matrix (in fact a simple list is used). This limits the necessary amount of memory per object. The sparsity of the co-occurrence matrix avoids its over population. Therefore, we have exactly one co-occurrence matrix per object trained and the final dimensionality of the training data representation is given by the squared number of cluster centers times the number of objects represented in the database.

For the recognition of objects with the co-occurrence matrices we follow in principle the queue for training but for a single query image. We calculate keypoints, compute the descriptors, identify the associated cluster indices by traversing the *k*-means tree and build the co-occurrence matrix for the query image. The matching procedure itself is deliberately kept very simple. We calculate the matching score for every object representation of the training database by a simple AND operation of the sparse co-occurrence matrices and counting the number of resulting matches. In fact, the matching is only a primitive maximum voting of congruent co-occurrences in the binary matrices.

4 Arbitration Strategy

The arbitration-component improves the results of the standard inverse voting approach with the additional information obtained by the co-occurrences. As we want to avoid any time consuming adaptation to a specific data set, we apply a heuristic algorithm providing good results.

The results of 'inverse voting approach' and 'co-occurrences' cannot be directly combined due to the completely different matching strategy. So we 'unify' the output to a very abstract layer. Each algorithm selects one object as distinct answer and provides a 'level of significance'. In particular we use three levels of significance: 'unambiguous', 'low confidence' and 'unknown' object. A simple set of rules is used to adjudicate on further processing.

```
object getFinalObjHypoth(InvVoting, CoOcc)
{

  object final;
  if(InvVoting.objHypoth == CoOcc.objHypoth)
  {
    // hypotheses agree
    final = InvVoting.objHypoth;
  }
  else if( InvVoting.confRate == 3 )
  {
    // inverse voting hypothesis is confident
    final = InvVoting.objHypoth;
  }
  else if( InvVoting.confRate > CoOcc.confRate )
  {
    // inverse voting is more confident than CoOcc
    final = InvVoting.objHypoth;
  }
  else if( CoOcc.confRate > InvVoting.confRate )
  {
    // CoOcc are more confident than inverse voting
    final = CoOcc.objHypoth;
  }
  else
  {
    // get decision from ranking of hypotheses
    final = RankingFunction(InvVoting, CoOcc);
  }

  return final;

}

object RankingFunction(InvVoting, CoOcc)
{

  ranking iVR = InvVoting.objHypoth.rankingInHistogram;
  ranking cOR = CoOcc.objHypoth.rankingInHistogram;

  if( iVR is better than cOR)
    // inverse voting hypothesis is chosen
    return InvVoting.objHypoth;
  else
    // cooccurrence hypothesis is chosen
    return CoOcc.objHypoth;

}
```

Figure 3: Pseudo-Code of our arbitration-module.

If the inverse object voting algorithm shows 'unambiguous' confidence, the proper object is used as final selection. If the inverse object voting algorithm shows lower confidence ('low confidence', 'unknown') the result obtained by the co-occurrence vote is taken into account. A special handling is required if both approaches have the same (low) level of significance but vote for different objects. In this

case, we search for the ranking of each selected object in the other algorithms ranking cue. Finally the object with the prior ranking is selected. A pseudo-code of the arbitration-module is depicted in Figure 3.

4.1 Level of significance for inverse object voting

The significance value of the voting histogram obtained by the inverse object voting algorithm could easily be evaluated by computing the ratio between the number of votes of the first and second ranked objects. The lower the ratio, the higher is the difference between those two ranked objects and therefore the significance score will be set higher.

$$k = \frac{x_2}{x_1} \tag{2}$$

where x_1 is the number of votes of the top ranked object and x_2 the counts of the second one.

$$\left\{ \begin{array}{l} k < t_2 \Rightarrow `unambiguous' \\ t_2 <= k <= t_1 \Rightarrow `low\ confidence' \\ k > t_1 \Rightarrow `unknown' \end{array} \right\} \tag{3}$$

Best results could be achieved if the thresholds are set as follows: $t_1 = 0.6$ and $t_2 = 0.5$.

4.2 Level of significance for co-occurrences

The level of significance for co-occurrences is determined by two facts. The first one is an absolute threshold, which assigns all objects with less than t matched co-occurrences to the 'unknown' confidence level (typically $t = 5$). The second one is related to the 'peakedness' of the voting histogram for co-occurrences. As a quantitative estimate for that, we calculate the kurtosis of the discrete voting histogram function. The kurtosis k of N discrete samples is a statistical measure for the 'peakedness' of a function and is defined by

$$k = \frac{1}{N} \sum_{j=1}^{N} \left(\frac{x_j - \overline{x}}{\sigma} \right)^4 \tag{4}$$

where x_j is the j-th sample (voting histogram entry), $\overline{x}$ the mean of the samples and σ the standard deviation of the samples distribution.

As the 'ideal kurtosis' (impulse function) for a perfect voting histogram is proportional to the number of objects (discrete samples in histogram), we normalized the kurtosis. Thus we can choose the decision thresholds for assignment to different confidence levels relative to the 'ideal kurtosis' (t_1, t_2).

$$\left\{ \begin{array}{l} k >= t_1 \Rightarrow `unambiguous' \\ t_2 <= k < t_1 \Rightarrow `low\ confidence' \\ k < t_2 \Rightarrow `unknown' \end{array} \right\} \tag{5}$$

In our experiments we obtained best results by setting $t_1 = 33\%$ and $t_2 = 10\%$.

5 Experiments

For the object recognition task we took a subset of 400 objects from the publicly available Amsterdam Library of Object Images (ALOI) [8] (see Figure 5 for some examples). The ALOI database consists of 1000 different objects captured on a turntable with 5 degree viewpoint changes and different illumination conditions. We primarily selected the objects according to the requirement to have a sufficient number of DoG keypoints detected on the objects surface.

To obtain visual words we detected DoG-points and calculated SIFT descriptors in only a subset of 100 labeled objects presented in 7 different views (700 images from -90^o to 90^o in steps of 30^o) and performed an agglomerative pre-clustering. After that we generated a k-means vocabulary tree with a branch factor of $k = 9$ with about 140.000 visual words received from the previous pre-clustering process.

In order to capture enough variances in the appearances of an object during the training stage, we presented 5 views of 400 objects to the system (2000 images from -60^o to $+60^o$ in steps of 30^o). Descriptors of the first 100 objects were included in the training set of the vocabulary tree, but 300 objects presented totally new feature descriptors. To evaluate the recognition rate we took 13 views from 400 trained objects (from -60^o to $+60^o$ in steps of 10^o) and performed queries with 5200 sample images.

We investigated the performance rates of this vocabulary tree and its inverted file structure in the following experiments. The first experiment shows the better recognition rate obtained by our efficient arbitration strategy with different numbers of clusters used by descriptor co-occurrences. After that we demonstrate the robustness of our approach to background clutter and occlusion.

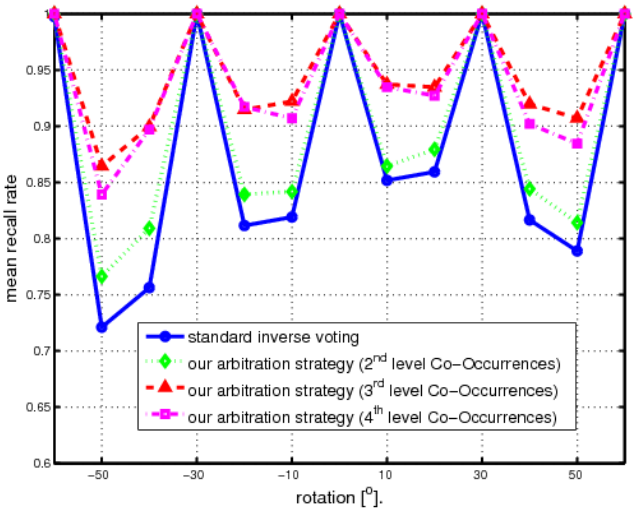

Figure 4: Comparison of different recall rates for different levels of co-occurrences. Using only a little amount of additional information from the second level in the vocabulary tree results in an average performance increase of about 4%. While accessing information from the third level leads to an overall increase of up to 12% there is no further performance improvement when taking into account information from a higher level (fourth level results in about 11%).

5.1 Performance comparisons

In this experiment we investigate the influence of additional information from different levels of co-occurrences on the

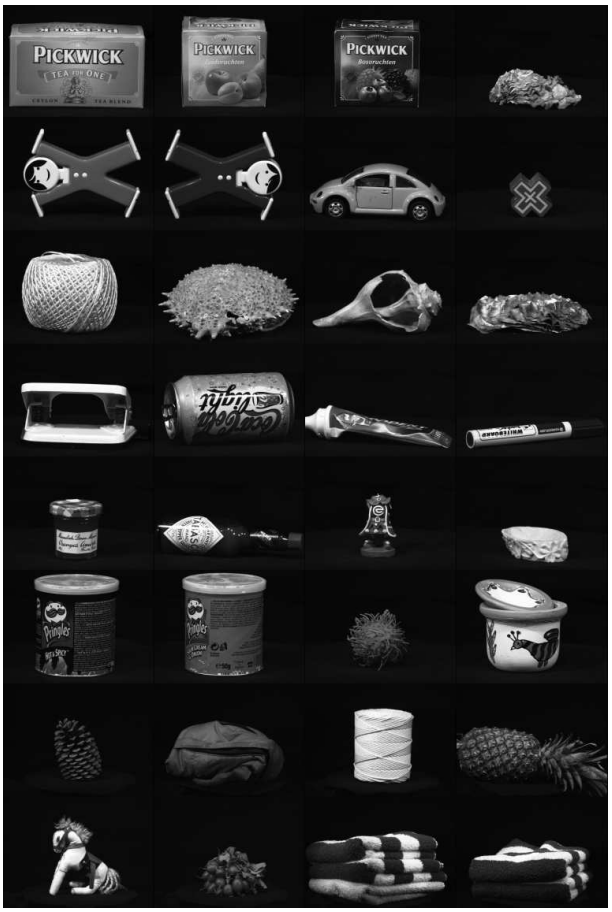

Figure 5: 32 sample images from 400 selected objects from the ALOI database used in our object recognition system. To illustrate the challenge, we have selected a number of very similar objects which might easily be confused, as well as a number of very complex objects which are rather hard to capture in an object representation.

final recognition performance. In Figure 4 the results of our method and the pure inverse voting result on the whole rotation range of 120 degrees are shown to demonstrate the power of our approach. While the blue curve describes the results from the standard inverse voting, the other curves are obtained incorporating information from three different levels of co-occurrences into our final reasoner. An overall increase of about 4%, 12% and 11% can be observed using information from levels two, three and four. There is a performance maxima using co-occurrence information from level three, which means that a number of $9^3 = 729$ co-occurrence cluster centers is considered. At the same time this means that the abstract information extracted from this level generalizes best for this object recognition setup. On the one hand a level of two results in an 81x81 co-occurrence matrix which is small and thus too noisy to gain a significant advantage from. On the other hand, a level of four produces a 6561x6561 matrix, which is taken about two levels above the leaves of the tree.

In Figure 6, four query examples, the intermediate results and the final object hypotheses are depicted. The first three query objects are labeled correctly (indicated by a green border) by our arbitration strategy although one of the intermediate object hypotheses is incorrect (red border). In the first two cases, the use of co-occurrences enables our approach to drawing the right decision even if the inverse voting chooses the wrong object. In the third case the arbitration-module still correctly favors a strong inverse voting result over a weak co-occurrence voting. The last object is labeled incorrectly, but note that the query object is a member of the group of objects chosen to be the correct hypothesis. Though the representation of the query object is a real subset of the representation of the hypothesis chosen, such an outcome is still registered as an error in our performance evaluation.

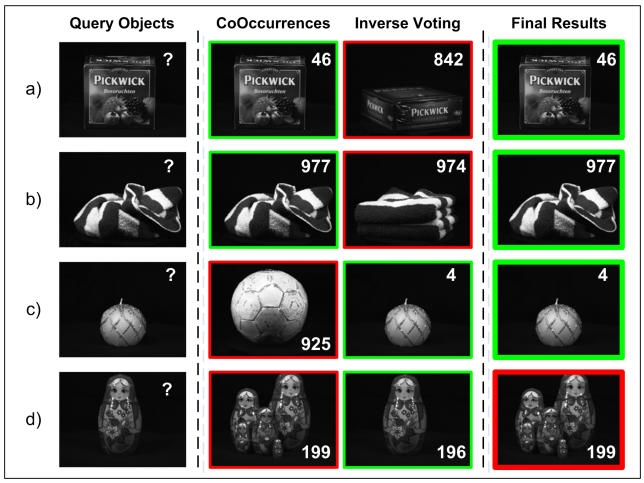

Figure 6: Four sample query images, the intermediate outcomes and the final results of our arbitration strategy are shown from left to right. While correct hypotheses are marked with a green border, a red border indicates a wrong outcome.

5.2 Substantial cluttered background images

For a realistic object recognition scenario, we projected each object onto a set of complex background images. Figure 8 depicts the mean recall rates for 400 objects on substantial cluttered background images using the segmentation masks provided in [8] (see Figure 7 for some examples). Please note, that the background images are especially challenging due to the high number of interest points obtained on the background. Thus the number of descriptors calculated on the background part is almost a multiple of that obtained on the objects surface. It is natural that the averaged recall rate is not as high as in the ideal case with segmented objects. However, for query images not seen during training an average increase in recognition rate of about 9% can be observed.

5.3 Results obtained by occlusion

To support the claim on the robustness of the obtained performance increase for occluded objects, we made some experiments with varying partial occlusions of the objects in recognition. We simulate the occlusions by removing a substantial part of the objects keypoints (similar to a black rectangle applied on the image). As the objects of the ALOI database are not normalized with respect to their appear-

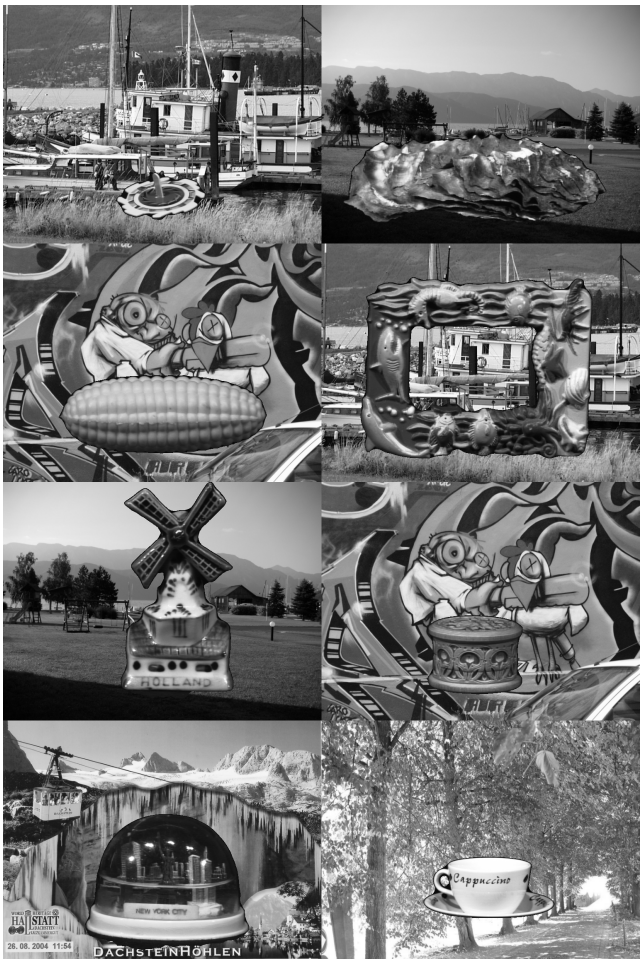

Figure 7: 8 sample images from 400 selected objects from the ALOI database projected on different backgrounds.

ance size (objects are of completely different size), a relative occlusion with respect to the full image dimensions would penalize small objects. It can happen, that such objects 'disappear' completely as no more keypoints are left even for a small amount of total image occlusion. Therefore we determined the relative area of occlusion for each object separately and with respect to the lateral cut of the particular object observed.

Figure 9 shows the mean recall rates for a different amount of occlusions. The mean recall rates for the (standard) inverse voting approach and our combined approach (arbitration strategy) remain rather stable up to an occlusion about 40% showing the robustness of local approaches to substantial occlusions. The performance increase by our combined approach is also constant about 8-10% with respect to the standard approach for all tested occlusions.

A motivation for the usage of weaker descriptors comes from the fact, that the recognition speed of the current implementation is limited by the runtime of the already highly optimized C/C++ implementations of keypoint detection (DoG) and their descriptors (SIFT) [12] while our approach is currently implemented in MATLAB. Table 1 gives a raw overview about the relative runtime effort spent in different components of the recognition stage. Only 17.4% of

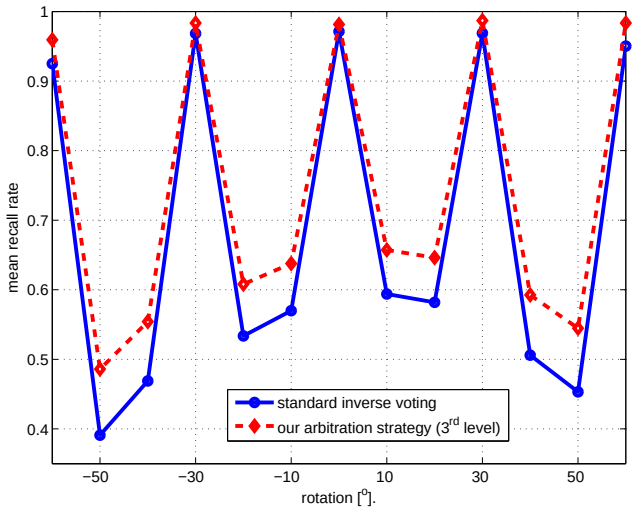

Figure 8: Mean recall for objects projected on a set of substantial cluttered background images.

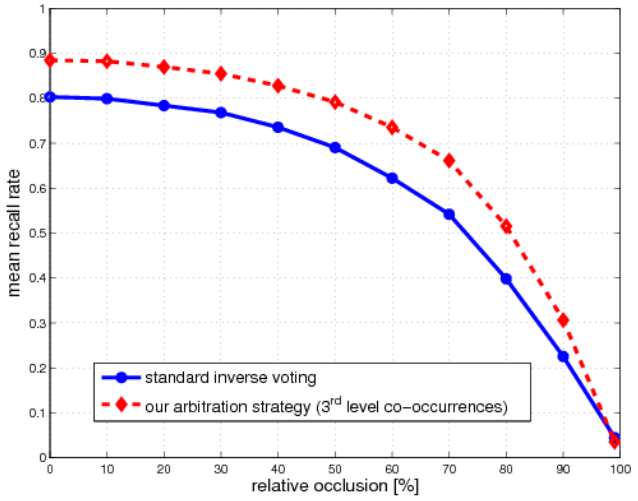

Figure 9: Mean recall rates of the inverse voting approach and our arbitration strategy for a different amount of occlusions.

the overall burden for recognition is used for the assignment of obtained query-features to the corresponding cluster centers by the *k*-means tree. The computational costs for the voting and arbitration strategy parts are nearly negligible. So reducing the calculation effort for keypoint detection and descriptor calculation by coevally obtaining high recognition performance would further improve the efficiency of our approach.

6 Conclusion and Future Work

In this paper we have introduced a new method to increase the recognition performance of a vocabulary tree based recognition system. We improved the hypotheses of an inverse object voting algorithm by a very simple specificity of spatial relations, namely descriptor co-occurrences. A rather heuristic but powerful arbitration strategy with minimal computational effort amplifies the specific strengths

component	runtime (ms)	%
DoG and SIFT calculation	3351	80.6
assignment (tree propagation)	723	17.4
inverse voting	62	1.5
co-occurrences	15	0.4
arbitration strategy	4	0.1

Table 1: Mean runtimes of certain recognition components obtained on a Intel Xeon 2.80GHz CPU.

of the particular representations. The achieved increase of performance has been demonstrated in a challenging object recognition task and we have also shown the robustness of the approach even for a substantial amount of occlusions and cluttered background.

The main advantage of our approach is the fact, that we use two different levels of information abstraction provided in various depths of the tree. Thus we can avoid the calculation of an additional, computational representation for the post verification step. As the main computational burden of the recognition system is carried by calculation of the keypoints and their descriptors, in future research we will use our approach to work with even weaker detectors and descriptors but keeping recall rates high by combination of two or more levels of information abstraction.

References

[1] Ankur Agarwal and Bill Triggs. Hyperfeatures – multilevel local coding for visual recognition. In *Proceedings 9th European Conference on Computer Vision*, volume 3951 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2006.

[2] Guillaume Bouchard and Bill Triggs. Hierarchical part-based visual object categorization. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 710–715, 2005.

[3] Gustavo Carneiro and David Lowe. Sparse flexible models of local features. In *Proceedings 9th European Conference on Computer Vision*, volume 3953 of *Lecture Notes in Computer Science*, pages 29–43, 2006.

[4] David J. Crandall and Daniel P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *Proceedings 9th European Conference on Computer Vision*, volume 3951 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2006.

[5] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.

[6] Rob Fergus, Pietro Perona, and Andrew Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[7] Friedrich Fraundorfer and Horst Bischof. Evaluation of local detectors on non-planar scenes. In *Proceedings 28th Workshop of the Austrian Association for Pattern Recognition*, 2004.

[8] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W. M. Smeulders. The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[9] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Affine-invariant local descriptors and neighborhood statistics fortexture recognition. In *Proceedings 9th International Conference on Computer Vision*, pages 649–655, 2003.

[10] Bastian Leibe, Krystian Mikolajczyk, and Bernt Schiele. Efficient clustering and matching for object class recognition. In *Proceedings 17th British Machine Vision Conference*, 2006.

[11] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2:775–781, 2005.

[12] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[13] Krystian Mikolajczk and Cordelia Schmid. A performance evaluation of local descriptors. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 257–263, June 2003.

[14] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Multiple object class detecton with a generative model. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[15] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oktober 2005.

[16] Takeshi Mita, Toshimitsu Kaneko, and Osamu Hori. Joint haar-like features for face detection. In *Proceedings 10th International Conference on Computer Vision*, volume 2, pages 1619–1626, 2005.

[17] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[18] Stepan Obdrzalek and Jiri Matas. Sub-linear indexing for large scale object recognition. In *Proceedings 16th British Machine Vision Conference*, volume 2, 2005.

[19] Andreas Opelt, Axel Pinz, and Andrew Zisserman. Incremental learning of object detectors using a visual alphabet. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 3–10, 2006.

[20] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[21] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location in images. In *Proceedings 10th International Conference on Computer Vision*, volume 1, pages 370–377, 2005.

[22] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings 9th International Conference on Computer Vision*, pages 1470–1477 vol.2, 2003.