
Building Classification

Mingxuan Sun, Adebola Osuntogun and Matthias Grundmann
{cynthia, bola}@cc.gatech.edu, grundman@cs.tum.edu

Abstract

Object recognition and image classification has become an area of intense focus in machine learning and pattern recognition. Building recognition and classification rise into attention in the area of urban reconstruction where a number of images of buildings are used. We first compare different classification approaches to discriminate the class *building* from the class *non-building*. The images are obtained from the Caltech256 database. Secondly we recognize individual buildings in the ZuBuD database, consisting of 201 individual buildings. We compare three different algorithms: Consistent Line Clusters (CLC), Randomized Decision Trees and the Vocabulary Tree. We obtain 99% accuracy on the *building* versus *non-building* classification and 75.6% on specific building classification. We conclude that the CLC method performs best in first case, while the Vocabulary Tree performs best in the second case, although Randomized Trees perform similar (72%).

1 Introduction

Building recognition and classification has several motivations in the realm of urban reconstruction and city navigation. In the 4D City reconstruction project managed by Georgia Tech, tons of pictures obtained from either online or a library are classified into building and non-building images. The building images are further classified so that images belonging to the same individual class will be selected in 3D reconstruction for a specific building model.

Specific building recognition can be commercialized and used on cell phones to guide tourists through a city like an outdoor museum. One can imagine a mobile application in which the user takes an image of the building and is able to obtain information about the history of the building including when it was built and maybe modifications that occurred over time, interesting events that have occurred in the building, et cetera. Such ubiquitous computing applications and others serve as the motivation for our work.

The challenge is to construct methods that are not only adaptive to variations but scale well with the size of the database. We explore the impact of varying the number of images for each class to the classification accuracy in the case of discriminating *buildings* from *non-buildings*. In the case of recognizing individual buildings we vary the number of images for each building to determine its impact on the results.

We use the following image databases:

ZuBuD database [1] consists of images of 201 buildings captured from five different viewpoints in Zurich city and 115 query images taken from different viewpoints under varying lightening conditions. This database is widely used for classification [18, 7, 5, 20].

Caltech 256 database [2] consists of 256 image categories including cars, animals, skyscrapers etc. We focus on distinguishing skyscrapers (95 images) from other categories.

We explore three different supervised learning algorithms. First we use the specialized approach of recognizing consistent line clusters (CLC) to differentiate *buildings* from *non-buildings* as described in [6]. It uses the assumption of parallelism among elements that form buildings (e.g. symmetric

placements of windows). Second we recognize individual buildings by using decision trees with random subwindows [19] and by building a vocabulary of visual words that is efficiently represented in a vocabulary tree [17].

The Caltech 256 database is used to distinguish *buildings* from *non-buildings*. Images for the *non-building* class are randomly chosen over the other remaining 255 categories. All three algorithms described above are applied to this dataset and their performance is compared.

The ZuBuD database is used to recognize individual buildings. Classification results obtained by randomized decision trees and the vocabulary tree are compared.

Using these three algorithms we want to achieve competitive classification and recognition results, in comparison with the cited literature. The recognition error for *buildings* versus *non-buildings* achieved by [6] is 5.8%, although the algorithm was run on an independent database. The recognition rates for recognizing specific buildings on the ZuBuD database range from 41% [20] to 93% [18]. Our goal is to achieve recognition rates in this range.

2 Problem setup

We first run several experiments to calculate a general baseline for comparison of our algorithms, and to obtain some insight about the nature of the data we are using. We use four different naive classifiers:

Majority classifier We constrain the datasets to have equal sized classes, so the algorithms should perform better than random guessing, that means better than $\frac{1}{\#classes}$. That leads to lower bounds of 50% for the Caltech256 and 0.49 % for the ZuBuD dataset.

Mean color of image Each test image is classified by selecting the closest image in the database, defined by the Euclidean distance in \mathbb{R}^3 (RGB)

Histogram comparison A histogram for each image is computed and each test image is classified by selecting the closest histogram in the database. Histograms are normalized w.r.t. sum of all pixel intensities to obtain invariance under different lighting conditions. Comparison is done by computing the discrete Kullback-Leibler-Divergence between two histograms h_1 and h_2 : $\sum_{i=0}^{255} h_1(i) \log \frac{h_1(i)}{h_2(i)}$

k-NN We compute MSER features and descriptors (described below) for all images. We classify a test image by using k-NN to classify its features descriptors and take the majority vote.

Furthermore we investigate how well the algorithms scale (i.e. accuracy and run-time) with the size of the classification problem. In the case of *building* versus *non-building* we run each algorithm on 25%, 50% and 100% of the 95 images for each class to investigate the accuracy w.r.t. the database size. In the case of recognizing individual buildings we limit the number of individual buildings in the train and test set to 25%, 50% and 100% of the 201 different buildings.

3 Algorithms description

3.1 Consistent Line Clusters

There are many Content Based Image Retrieval (CBIR) methods that are similar to Consistent Line Clusters (CLC), for example the approach implemented in [19] uses perceptual grouping of rectangles for building recognition. In [22] edge-map features are extracted and [4] apply edge-direction-histogram (EDH) features to discriminate between city versus landscape images. We implement CLC [6] and test it on classifying building versus non-building images, using the C4.5 package to generate decision trees and perform cross validation.

To recognize buildings, lines play a very important role. Symmetrical and parallel arranged elements in the 3-D space, like windows, doors and the outside shape of the building form horizontal and vertical lines, which can be used for recognition. The first step of the algorithm is line cluster detection. Figure 1 shows our detection result for *buildings* versus *non-buildings*. Guided by vanishing points in the image, the lines are clustered into different groups by their spatial orientation. For images

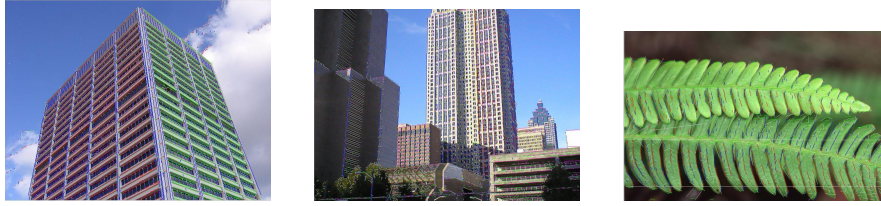


Figure 1: Left: line features for single building, Middle: for multiple buildings, Right: line features for non-building

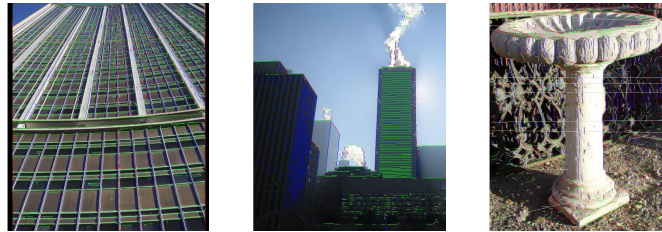


Figure 2: Left: strong inter-relationship, Middle: strong intra-relationship, Right: non-regular lines

with a single building, as shown in Figure 1, there are three major directions (three vanishing points) labeled as red, green, and blue respectively. For images with multiple buildings, more line clusters are detected due to more vanishing points.

The second step is to compute feature descriptors for the image from its line clusters. According to the algorithm, we use two criteria to detect buildings: the inter-relationship of clusters represents structure-preserving or junction-rich buildings, the intra-relationship represents simple-structured or overlap-rich buildings. More specifically the inter-relationship criterion computes the maximum number of line intersections for each line cluster and the intra-relationship criterion examines how many lines overlap within a line cluster. As shown in figure 2 building images show either strong inter-relationship features or intra-relationship features while non-building images do not.

3.2 Extremely Randomized Decision Trees

Decision tree ensembles have been shown in [11] to perform fairly well in different classification tasks and are relatively easy to implement. We will explore classification of building images using decision tree ensembles. They achieve better results than classical ones which suffer from high variance. In [11] a new algorithm was developed for image classification known as extremely randomized trees (extra trees). The algorithm for building the extra trees uses top down induction, tests are performed at internal nodes that compare the value of a pixel to a threshold. This algorithm uses the pixel values of the images for classification and so does not require extraction of any features as most image classifications algorithms do. At each node, a random pixel location is chosen for all images and the mean and standard deviation of the pixel value's distribution over all images is used to split the set of images and grow the decision tree. It differs from the classical decision tree algorithm because it uses randomization to split without using any score measure. The running time for this algorithm is $O(N \log N)$, where N is the number of images. A description of the algorithm is given in [10]. Figure 3 shows in red the pixel locations that are selected for testing this image as it is propagated through the tree. A large number of pixel locations are selected as can be seen in the image.

3.3 Vocabulary Tree approach

In order to increase the number of object classes that can be recognized and discriminated we follow the approach described in [17]. The approach extends the idea of [21] and [23] of building a vocabulary of visual 'words' over a large set of images and solely relies on these words for further



Figure 3: Sample images with pixel locations

recognition. This does not only reduce the computational complexity, but also increases generalization, because similar information is discarded. The words are highly-distinctive descriptors evaluated at feature points in an image. The feature points in the original implementation [17] are given by the center of Maximally Stable Extremal Regions (MSER) [12]. At these feature points SIFT descriptors [8] are extracted and quantized with a vocabulary tree, described in detail below. The MSER is an affine invariant region detector, which has been found to perform superior to many other affine region descriptors in the task of matching [16]. However their performance to generalize beyond learned images in the task of object recognition is not as good as Hessian-Affine detectors (HA) [13], which are described in [14]. On account of this we examine the difference in classification accuracy between the MSER and the HA detector and evaluate whether a combination of both can increase the recognition rate, as indicated in [21].

The vocabulary tree is basically a hierarchical application of k-means. First, a basic k-means is applied to the descriptor vectors, resulting in k clusters. Second, k-means is recursively applied to each of the k clusters, leading to a tree-structure of nested quantization cells. The process is stopped when a maximum level L is reached. Trying to find the nearest cluster of a query point is linear to $k \cdot L$, while a normal k-means with the same number of cluster centers can take in worst-case k^L evaluations, even when implemented efficiently with a k-d tree.

For each image in the training set a vector d describing the image is created, which has an entry for each node in the tree ($\sim k^L$). The entry d_i of d is the number of descriptors of the image passing through the node i weighted by an entropy-like factor specifying the importance of the particular node. The same principles are used to compute a query vector q for a test image. A test image is classified by selecting the closest image from the database that minimizes $\|q - d\|$.

4 Classification Results

4.1 Naive Classifiers

In order to obtain a baseline for our classification algorithms we used classification by mean color, histogram similarity and k-NN for feature vectors as described above. The results of classifications are given in table 1. Because of the high diversity among the images in the Caltech256 skyscraper subsection the mean color algorithm performs very bad, while its result on the ZuBuD dataset is good. This leads to the conclusion, that the mean color of the buildings in the ZuBuD dataset is already a discriminative feature. The histogram algorithm performs poor in both cases, the distribution of pixel intensities is not an discriminative feature in both datasets. The kNN algorithm achieves also poor results on the Caltech256 dataset, presumably due to its high inner-class variation, while it performs very good on the ZuBuD dataset. So the extracted feature-descriptors are very discriminative and model the specifics of an individual object very well. In the case of correct classification, the k-NN algorithm labels 77% of the feature vectors in the 2-classes case and 39% ($\gg 0.49\%$) of the feature vectors for the 201-classes case correct. This explains its goods performance on the ZuBuD dataset, although it takes a prohibitive computation time for a real application.

4.2 Consistent Line Clusters

The feature vectors are computed from the inter- and intra-relationship of line clusters. They are used to generate simple decision-tree classifiers. We use the C4.5 package for this task. We select 95 building images and 95 non-building images from the Caltech256 dataset and run a set of cross-validation experiments in which 90% of the images are used as the training set. The results are showed in table 2 and compared with neural network and boosting of the decision tree. Results of

| DataSet | Classes | Algorithm | Classification Time (s) | Accuracy |
|------------|---------|------------|-------------------------|----------|
| Caltech256 | 2 | Mean Color | 0.45 | 9.57 |
| Caltech256 | 2 | Histogram | 0.54 | 53.91 |
| Caltech256 | 2 | kNN | 6509 | 67.9 |
| ZuBuD | 201 | Mean Color | 1.2 | 58 |
| ZuBuD | 201 | Histogram | 10.3 | 20 |
| ZuBuD | 201 | kNN | 56880 | 88.89 |

Table 1: Classification Results for naive Algorithms



Figure 4: Two left: false negative due to poor lighting condition or poor pattern; Two right:false positive due to good structure

misclassifications are shown in figure 4. The two examples on the left show false negative cases when there are not enough patterns to detect. The two examples on the right show false positives when there are well-structured objects like fences.

| Dataset | Scale | Classes | Algorithm | Accuracy | False-pos. | False-neg. | t_{train} | t_{test} |
|------------|-------|---------|---------------|----------|------------|------------|-------------|------------|
| Caltech256 | 0.25 | 2 | Decision Tree | 0.96 | 0.08 | 0.00 | 0.00 | 0.00 |
| Caltech256 | 0.25 | 2 | Neural Net | 0.97 | 0.01 | 0.03 | 0.43 | 0.00 |
| Caltech256 | 0.25 | 2 | Boosted Tree | 0.99 | 0.01 | 0.00 | 0.00 | 0.00 |
| Caltech256 | 0.5 | 2 | Decision Tree | 0.98 | 0.02 | 0.02 | 0.00 | 0.00 |
| Caltech256 | 0.5 | 2 | Neural Net | 0.95 | 0.02 | 0.08 | 0.82 | 0.00 |
| Caltech256 | 0.5 | 2 | Boosted Tree | 0.97 | 0.01 | 0.03 | 0.02 | 0.00 |
| Caltech256 | 1 | 2 | Decision Tree | 0.99 | 0.00 | 0.01 | 0.00 | 0.00 |
| Caltech256 | 1 | 2 | Neural Net | 0.98 | 0.00 | 0.03 | 1.24 | 0.00 |
| Caltech256 | 1 | 2 | Boosted Tree | 0.995 | 0.00 | 0.01 | 0.03 | 0.00 |

Table 2: Classification Results for LineCluster algorithm

4.3 Extremely Randomized Decision Trees

Table 3 shows the results obtained by resizing the images to 50x50 pixels and creating several randomized decision trees. The accuracy is given for the specific building classification. Since the decision trees are randomly generated, we run the algorithm several times with a specific number of trees and then average the results. The table shows that as the number of trees increases, the error rate decreases and so we obtain better results. Given that this algorithm works on the raw gray level pixel values, these results are quite good because they do not require any form of feature extraction or prior manipulation of the image. Table 4 shows the results obtained by reducing the percentage of the training data available for both datasets. The results on the ZuBuD dataset decrease when all the data is used, probably because at each iteration there is a varying number of training and testing data so using all the data increases the chance of errors. On the Caltech256 dataset, there appears to be a fluctuation in the results obtained but the major trend indicates that as the training data increases less trees are needed for classification.

Analysis of the confusion matrix, indicates that similar buildings are being confused. For example images in figure 5 below are placed in the same class. Visually, these images have the same brick type or color scheme indicating that decision trees are learning the correct concepts.

| Number of Trees | Accuracy | TrainTime | TestTime |
|-----------------|----------|-----------|----------|
| 10 | 0.21 | 4.83s | 0.652s |
| 100 | 0.45 | 4.68s | 0.733s |
| 500 | 0.65 | 6.413s | 1.035s |
| 1000 | 0.68 | 6.642s | 1.461s |
| 2000 | 0.71 | 19.205s | 2.558s |

Table 3: Classification results for extremely randomized decision trees algorithm

| Data-Set | Scale | Number of Trees | Accuracy | TrainTime | TestTime |
|------------|-------|-----------------|----------|-----------|----------|
| ZuBuD | 25% | 1000 | 0.77 | 2.084s | 0.478s |
| ZuBuD | 50% | 1000 | 0.77 | 3.77s | 0.708s |
| ZuBuD | 100% | 2000 | 0.71 | 19.205s | 2.558s |
| Caltech256 | 25% | 2000 | 0.76 | 0.96s | 0.342s |
| Caltech256 | 50% | 1000 | 0.71 | 1.041s | 0.336s |
| Caltech256 | 100% | 100 | 0.73 | 0.774s | 0.341s |

Table 4: Classification results for scaled data

4.4 Vocabulary Tree

For the feature extraction stage used for kNN-Clustering and the Vocabulary Tree, we used the binaries available from [3] for detecting Hessian-Affine and MSER regions and computing GLOH-descriptors. GLOH-descriptors are similar to SIFT-descriptors but computed on a log-polar grid instead of a rectangular one and have been found to perform better [15]. The MSER detector produces around 500 features per image, while the Hessian-Affine produces around 1000 features per image, in both cases applied to the ZuBuD database. In order to determine the best parameters of the branching factor k and the tree depth, several experiments were run. The branching factor k does not seem to play an important role, as shown on the left of figure 6. In fact with increasing depth, i.e. increasing nodes in the tree, each $k \in \{3, 5, 10, 20\}$ achieves a classification accuracy around 74 %. The graph indicates, that $k = 20$ needs the fewest nodes to accomplish this. On the other hand figure 7 shows, that although $k = 20$ has the fewest nodes, the training and classification time are higher than for other choices. The figure indicates that $k = 5$ seems to be the best choice for classification algorithm w.r.t. classification and training time. In the following experiments we used a branching factor of $k = 10$ and a depth of 6, because this choice leads to fast training times and to the highest classification rate for the ZuBuD database of 75.6 % in 40s. While the algorithm performs well on the ZuBuD dataset, its performance on the *building* versus *non-building* problem is poor (66%). With respect to the performance of the k-NN classification of the problem (67.9%) it can be concluded that the feature vectors are well suited for recognizing individual objects but do not generalize very well beyond this scope. The right figure 6 shows how well the algorithm scales. It is interesting that on the ZuBuD dataset the accuracy increases with the amount of different buildings. We assume, that this is mainly due to the highly discriminative nature of the feature vectors. We also compared the different types of features and the combination of them. On the *building* vs. *non-building* problem, MSER features give 61% accuracy while HA performed slightly better (66%). A combination of both does not performed better (61%). This is according to the literature [13], HA features generalize better, while MSER features are better suited for matching tasks. That is why MSER features perform better on the ZuBuD dataset, while Hessian-Affine features perform



Figure 5: Similar Buildings

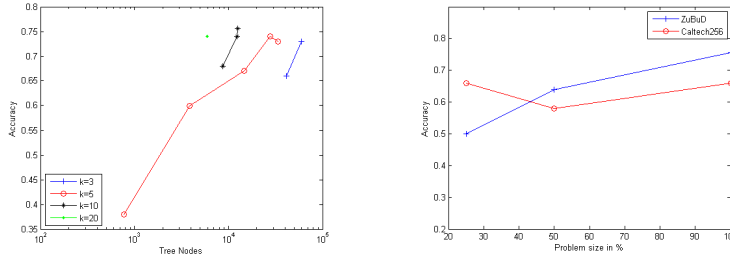


Figure 6: Left: Accuracy for different Tree Sizes for ZuBuD, Right: Accuracy for different problem sizes

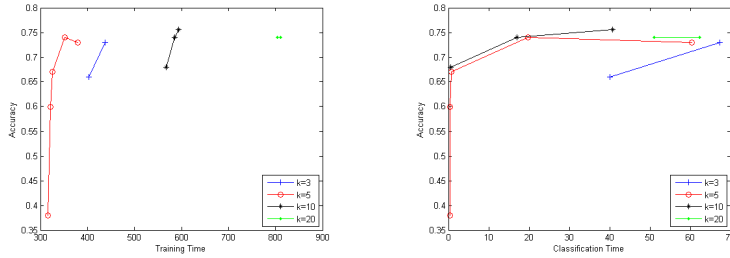


Figure 7: Left: Accuracy w.r.t. training time (s) for ZuBuD, Right: Accuracy w.r.t. classification time (s) for ZuBuD

around 1% worse. We also tested which effect the entropy-weighting for nodes has. Tested on the ZuBuD dataset the accuracy dropped to 36.4% when ignoring the entropy-weighting. We tried to give more weight to nodes farther away from the root, but this does not change the result more than 1%. Actually multiplying each weight by a random number $r \in (0, 1]$ also does not alter the results more than 1%. We conclude that the entropy-weighting is the most important and a very robust method to compare the vectors describing a train and test image.

Furthermore we tried to modify the original algorithm by dropping the fixed constraint of a specific depth but continue clustering until the cluster size is lower than a specific threshold. This could increase the accuracy by around 0.7 % on the ZuBuD dataset.

4.5 Conclusion

All algorithms show a nearly constant accuracy among different sizes of the data. Fluctuations can be explained with the relatively small size of our databases, so that subsets may include harder or easier to classify samples. While CLC are so fast, that we did not notice any difference in execution time among different scales, the Randomized Trees and the Vocabulary both need $O(N \log N)$, where N is the size of the database. This is a significant improvement over the exponential run-time of the k-NN algorithm.

Table 5 shows the best obtained accuracy and the training and classification time for the different algorithms and datasets including the best naive classifier. The CLC approach performs extremely good on the specific problem of discriminating *building* versus *non-building*, because it is especially designed for this specific task and its assumptions seem to be sufficient. Randomized Trees and the Vocabulary Tree perform worse, because they address more the problem of recognizing learned objects instead of discriminating object categories with a lot of inner-class variation. Although the vocabulary tree gives slightly better results on the ZuBuD dataset than the Randomized Trees, it takes longer to execute and is very memory intensive. For mobile application described in the introduction, the Randomized Trees should be preferred. It is important to note, that the naive k-NN classifier always outperform the Vocabulary-Tree. Due to its recursive nature we assume that the Vocabulary Tree acts *like* an approximation to k-NN, while performing more than 20 times faster.¹

¹The run-time advantage increases with the database size

| Data-Set | Algorithm | Accuracy | Training time | Classification time |
|-------------|------------------------------|----------|---------------|---------------------|
| Caltech-256 | CLC | 0.995 | 0.03 | 0.00 |
| Caltech-256 | Randomized Trees | 0.73 | 0.77 | 0.34 |
| Caltech-256 | kNN (MSER) | 0.679 | — | 6509 |
| Caltech-256 | Vocabulary Tree (Hess.-Aff.) | 0.66 | 39.11 | 0.49 |
| ZuBuD | kNN | 88.89 | — | 56880 |
| ZuBuD | Vocabulary Tree | 0.756 | 595.32 | 40.76 |
| ZuBuD | Randomized Trees | 0.71 | 19.2 | 2.56 |

Table 5: Classification results

References

- [1] <http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html>.
- [2] http://www.vision.caltech.edu/Image_Datasets/Caltech256.
- [3] <http://www.robots.ox.ac.uk/~vgg/software/>.
- [4] A.Vailaya, A.K.Jain, and H.J.Zhang. On image classification: City images vs. landscapes. In *Proceedings of Pattern Recognition*, volume 31, pages 1921–1936, 1998.
- [5] T. Goedeme, T. Tuytelaars, and L. Gool. Fast wide baseline matching for visual navigation. *CVPR*, 2004.
- [6] Y. Li and L. G.Shapiro. Consistent line clusters for building recognition in cbir. In *Proceedings of International Conference on Pattern Recognition*, 2000.
- [7] J.-H. Lim, J.-P. Chevallet, and S. Gao. Scene identification using discriminative patterns. In *ICPR '06*.
- [8] D. Lowe. Distinctive image features from scale-invariant keypoints. In *Int. J. of Comput. Vision*, 2003.
- [9] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Decision trees and random subwindows for object recognition. In *ICML workshop on ML Techniques for Processing Multimedia Content*, 2005.
- [10] R. Marée, P. Geurts, G. Visimberga, J. Piater, and L. Wehenkel. An empirical comparison of machine learning algorithms for generic image classification. In *23rd SGAI international conference*, 2003.
- [11] R. Marée, G. P., P. J., and W. L. A generic approach for image classification based on decision tree ensembles and local sub-windows. In *Proc. 6th Asian Conference on Computer Vision (ACCV)*, Jan 2004.
- [12] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, 2002.
- [13] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *ICCV '05*, pages 1792–1799, Washington, DC, USA, 2005. IEEE Computer Society.
- [14] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004.
- [15] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.
- [16] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of CV*, 2005.
- [17] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.
- [18] Š. Obdržálek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 1–10, London, UK, September 2005. BMVA.
- [19] Q.Iqbal and J.K.Aggarwal. Applying perceptual grouping to content-based image retrieval: Building images. In *Proceedings of Computer Vision and Pattern Recognition*, pages 42–48, 1999.
- [20] H. Shao, T. Svoboda, V. Ferrari, T. Tuytelaars, and L. J. V. Gool. Fast indexing for image retrieval based on local appearance with re-ranking. In *ICIP (3)*, pages 737–740, 2003.
- [21] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.
- [22] S.X.Zhou, Y.Rui, and T.S.Huang. Water-filling algorithm: A novel way for image feature extraction based on edge maps. In *Proceedings of IEEE Image Processing Conference*, 1999.
- [23] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1–2):61–81, Apr. 2005.