Performance Evaluation of Visual Vocabularies for Object Recognition

Michal Kottman[*]
Institute of Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
kottman@fiit.stuba.sk

Abstract

Visual vocabularies can be used in computer vision tasks including image classification and object recognition. Performance of methods which use visual vocabularies is dependent on multiple factors, such as the feature type used, vocabulary size and construction method. In this paper, we present an evaluation of visual vocabularies of varying sizes and four types of feature descriptors, including SIFT and SURF. We compare the performance of vocabularies constructed using k-means clustering and self-organizing (Kohonen) map. The performance of given clustering methods and descriptor types in terms of accuracy and speed is evaluated on our sign recognition method used for detecting hazardous signs in high resolution images.

Categories and Subject Descriptors

I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement—*feature representation*; I.5.2 [**Pattern Recognition**]: Design Methodology—*pattern analysis*; I.5.3 [**Pattern Recognition**]: Clustering—*algorithms*

Keywords

object recognition, feature descriptor, clustering, visual vocabulary, visual words

1. Introduction

Methods based on visual vocabularies (also called “codebooks”) are becoming increasingly popular in the field of computer vision, especially for object recognition tasks. The main principle of visual vocabulary is to construct a codebook of common image features, like corners, blobs or highly textured areas. These common features are then called *visual words* and are used for further processing instead of individual features.

The visual vocabulary is commonly constructed by quantizing the feature space using clustering algorithms into clusters. The centers of these clusters are the visual words. The quantization of feature space creates a metaphor, which allows natural text processing techniques to be used in computer vision tasks. Examples of such techniques are *bag of words* [12], *probabilistic latent semantic analysis (pLSA)* [4] and *latent Dirichlet allocation (LDA)* [8]. These methods are often used for image categorization (portrait, landscape, cars, ...) and object recognition.

We perform the evaluation on our object recognition algorithm, which uses the structure of pairs of visual words on a training image to locate and identify it in a target image. The algorithm supports the detection of an object under varying scale and rotation, which are determined using a generalized Hough voting process [1].

Our previous work (Sec. 2) resulted in the consideration of further possible improvements. Especially we see the potential in the use of other, more appropriate descriptors and other clustering methods, mainly in applying clustering methods which exploit the relationship between similar visual words. This led to the idea of applying self-organizing Kohonen map (SOM) in the process of visual vocabulary creation. This paper presents series of experiments which are based on the codebook generation by SOM compared with previously presented K-means clustering methods in combination with further descriptors like Daisy and “Normalized image patch”.

2. Object Recognition Using Pairs of Words

This section briefly introduces our object recognition method [3], which uses a visual vocabulary to identify visual words in an image and uses the structure of pairs of visual words to identify and locate a (possibly small) object in the image.

Our method was used to locate hazardous signs in an image for safety applications like traffic monitoring. Unlike other visual vocabulary based methods, our method concentrates on identifying different instances (individ-

[*]Doctoral degree study programme in field Applied Informatics. Supervisor: Assoc. Professor Martin Šperka, Faculty of Informatics, Paneuropean University in Bratislava. Work described in this paper was presented at the 8th Student Research Conference in Informatics and Information Technologies IIT.SRC 2012.

Kottman, M. Performance Evaluation of Visual Vocabularies for Object Recognition. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 4, No. 2 (2012) 38-42

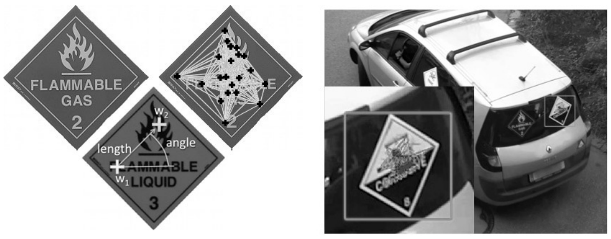

Figure 1: Our sign recognition method based on pairs of visual words. Left to right: an example sign; a pair of visual words w_1, w_2 with length and angle; the fingerprint of a sign; a sign is correctly detected on an image.

ual signs) instead of object category ("hazardous signs"). Therefore our method has to identify the differences between individual signs. The signs may be arbitrarily scaled and rotated in the image.

We use pairs of visual words to learn the structure of every sign (essentially creating a fingerprint), and then use pairs in the image to identify possible scale and rotation of the sign in the image. We then use this information to select relevant word pairs and use them to locate the sign. For every pair of visual words in the training sign, we store the words w_1, w_2, their distance d and angle α. To detect the sign, a two-stage voting process based on Hough transform is used. An overview of the process is illustrated in Fig. 1.

The first stage of the process is to determine a possible rotation and scale of the sign in the image. First, feature points are located in the query image and are described using the feature descriptor. Then they are assigned words using the visual vocabulary. Given each pair of visual words in the target image, a corresponding training entry (w_1, w_2, d, α) is looked up based on (w_1, w_2). The remaining values (d, α) are used to construct relative coordinates for the voting process. In our method, we only consider scale ratios $[0.5-2]$ times the training image size, and rotations of $\pm 30°$. Word pairs that are outside this range are discarded. Pairs that match the criteria then vote in a Hough accumulator for a concrete scale and rotation. The reasoning behind this voting process is that the relative rotation and scale will be almost the same for all pairs belonging to the sign. The highest value in the accumulator represents the most common transform.

Pairs that vote near the winning transform are passed to the second stage of Hough voting based on their position. The reason for this is that there may be ambiguous matches, but matches related to the sign are located close together. Furthermore, only the centroid of these pairs vote, because the centroid will be closer to the center of the sign. A sign is identified as detected if enough pairs pass the final voting. The voting space is visualized in Fig. 2. The levels of gray represent the number of votes for the given parameters.

Our method employs an "early exit" for sign detection – if not enough visual word pairs pass any stage, the sign is marked as not detected and any further processing is abandoned. This is an optimization which increases performance with bigger vocabularies, because they are more specific and rule out irrelevant pairs earlier.

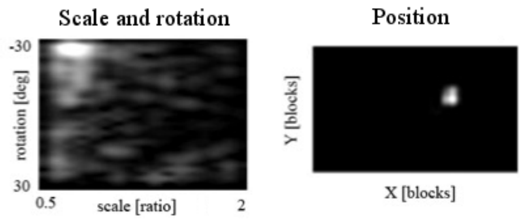

Figure 2: Detail of the Hough voting space. Black represents no votes, white represents the most votes. Left: scale and rotation phase based on quantized rotations (by 1°) and scale (by 0.05 scale points). Right: determination of the position of the sign using selected points. Space is quantized into blocks whose size is the detected scale of the sign.

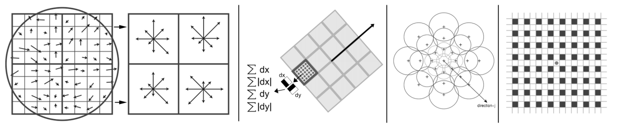

Figure 3: The feature descriptors evaluated in this paper. Left to right: a) SIFT descriptor b) SURF descriptor d) Daisy descriptor c) Normalized image patch.

3. Evaluated Local Feature Descriptors

In this paper, we evaluate four floating point feature descriptors. These descriptors transform image features into vectors of floating point numbers. The descriptors are used to describe the image features (preferably) invariant to various distortions, like rotation, scale change and illumination changes. The descriptors are illustrated in Fig. 3.

3.1 Scale-Invariant Feature Transform

The first evaluated descriptor is the *Scale-invariant feature transform (SIFT)* [7] (Fig. 3a). The algorithm consists of a feature detector and a feature descriptor.

The feature detector uses *Difference of Gaussians (DoG)* to identify local extrema in scale space, which are usually highly repeatable image features. The detector identifies the position (x, y), scale and rotation of the feature. The descriptor then calculates the histogram of gradient orientations in a 4×4 bins of oriented grid surrounding the feature point. Values are then normalized. In each bin, 8 orientations are calculated, which means that the resulting descriptor is a vector of 128 floating point numbers.

To evaluate the performance in a controlled environment, the image features detected by the *DoG detector* of the SIFT algorithm were used for every other description algorithm. We use a GPU implementation of the SIFT descriptor[1] to increase efficiency of the description,

3.2 Speeded-Up Robust Features

Speeded-Up Robust Features (SURF) [2] is a fast alternative to SIFT algorithm, and uses several speed enhance-

[1]SiftGPU: A GPU Implementation of Scale Invariant Feature Transform, http://cs.unc.edu/∼ccwu/siftgpu

ments over SIFT. It uses a *Fast-Hessian* detector based on integral images in detection phase, which allows it to find interest points quicker. In description phase, it does not calculate histogram of orientations, but uses simple sums of pixel values (Fig. 3b). The original descriptor provides 64 floats, but in our experiments, we use GPU implementation of an extended 128-element version of the descriptor, available in the OpenCV[2] library.

3.3 Daisy

The *Daisy descriptor* [11] (Fig. 3c) uses orientation histograms in a way similar to SIFT, but instead of using a rectangular grid, it uses a circular kernel of varying size with Gaussian weighting. It has been successfully used for wide-baseline stereo, and its shape has been shown to outperform state-of-the-art descriptors like SIFT and SURF, while being faster to compute than SIFT. Unlike other descriptors in this evaluation, no GPU implementation was available. The parameters of the descriptor have been altered to create a 128-element descriptor to match the previous descriptors.

3.4 Normalized Image Patches

To compare the descriptors with intuitive clustering based on image patches, we created a simple feature descriptor which describes image patches. The structure of the descriptor (Fig. 3d) is based on a grid of 8×8 pixel locations around the feature point. This layout has been inspired by *Histogrammed Intensity Patches (HIP)* [10].

In order to attain a level of invariance, the scale of the feature detected by the DoG detector is used to scale the patch appropriately. Instead of using binary vectors like in HIP, the values of the patch descriptor are transformed to a floating point number. The descriptor is then linearly normalized to range $[0, 1]$ using the minimum and maximum value in the grid. Apart from other descriptors in this evaluation, this descriptor is only 64 elements long.

4. Clustering

The visual vocabulary is created using a clustering method to partition the feature space of the training images. We compare the performance of Lloyd's algorithm [6], commonly also known as k-means clustering, and a self organizing map [5], also known as Kohonen map.

Both evaluated methods take the number of clusters as an input. Because the appropriate number of clusters is not known beforehand, we experimented with vocabulary sizes of $200, 500, 1000, 2000$ and 5000 clusters.

4.1 K-means Clustering

The main objective of the k-means clustering algorithm is to minimize the distance within a cluster (1). We have used squared Euclidean distance. C is a set of N clusters $C_1, \cdots, C_N \in C$, items $\vec{d_j}$ are the feature descriptors in the dataset and $\vec{c_i}$ represents the centroid of cluster C_i.

$$\arg\min_{C} \sum_{i=1}^{N} \sum_{\vec{d_j} \in C_i} \left\| \vec{d_j} - \vec{c_i} \right\|^2 \tag{1}$$

[2]Open Source Computer Vision library, http://opencv.willowgarage.com/wiki/

Because we are dealing with large amounts of high dimensional vectors, we used a parallelizable variant of the algorithm called h-means [9]. We implemented the algorithm for GPU using OpenCL[3]. Our implementation can process one iteration of clustering the entire dataset into 1000 clusters in less than one second.

4.2 Self Organizing Map

A self organizing map (SOM) is a form of vector quantization technique which discretizes the feature space onto a one or two dimensional map. Apart from k-means clustering, SOM attempts to preserve the topological properties of the feature space, mapping close features in a neighborhood on the map.

During the training phase, a feature vector $\vec{d}$ is randomly picked from the training dataset and a closest ("winning") map node is found. Then, the node and its neighborhood is moved closer to the input feature vector according to (2). In the equation, $\vec{m_i}$ represents the node being updated, $\alpha(t)$ represents time-decaying learning rate and $\Theta(t)$ is a neighborhood function based on the the distance of $\vec{m_i}$ from the winning node in the map.

$$\vec{m_i}(t+1) = \vec{m_i}(t) + \alpha(t)\Theta(t)\left[\vec{d} - \vec{m_i}(t)\right] \tag{2}$$

In our experiments, we used a two-dimensional rectangular map, created such that it contains the same number of nodes as for the k-means clustering - we used maps of sizes $20 \times 10(200), 25 \times 20(500), 40 \times 25(1000), 50 \times 40(2000), 100 \times 50\ (5000)$. A Gaussian neighborhood function was used, which was initialized to a radius of half the width of the map decaying exponentially with time, and the learning rate was set to 0.1 also with an exponential time decay.

The training was performed using 5 passes over the entire dataset. Although it is usually recommended that more passes over the dataset should be made, our experiments with 10 passes led to 2–5% decrease in accuracy.

5. Dataset

Visual vocabulary methods derive the vocabulary from a pre-determined set of training images. In this paper we use our own dataset of 175 Full HD (1920×1080) images used in a context of hazardous sign detection. The signs are very small relative to the whole image (usually around 100×100 pixels), but our method (see Section 2) can identify them.

As was previously stated, we use the SIFT algorithm to find interest features in the dataset. Because of the high resolution of the input images, the number of features found is large (between 2,000 and 5,000 features per image). To decrease the amount of processed features, we use the MSER algorithm as a preprocessing step to find regions of interest (see [3] for details). It reduces the number to 950 feature points per image on average. After preprocessing, the dataset contains 168,351 distinct features. The dataset also contains 11 different signs, with total 694 features.

[3]The implementation is available upon request.

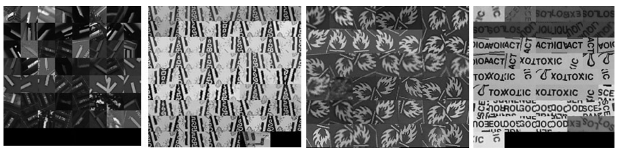

Figure 4: Examples of visual words extracted from our dataset. The images show compositions of patches from multiple images belonging to the same cluster. Words 1–3 show high consistency between images, while the last word shows ambiguous clustering.

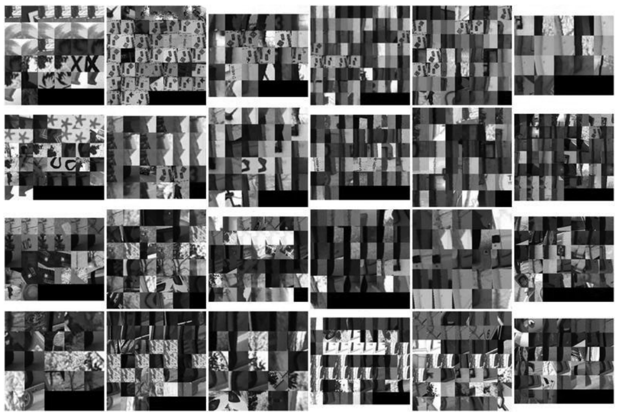

Figure 5: Example of 6×4 **adjacent nodes in the self organized map. In each node representing a visual word, patches from the dataset belonging to the word are displayed.**

An example of visual words in a vocabulary constructed from our dataset is shown in Fig. 4.

6. Evaluation

The evaluation phase can be separated into two phases – training and detection. During the training phase, the visual codebook is calculated from the feature descriptors in the dataset and a fingerprint is calculated for each sign. During the detection phase, visual word pairs are filtered as described in Sec. 2. If the number of resulting pairs is above a threshold, the given sign is detected as found in the image.

6.1 Training

1. Select a feature descriptor
2. Calculate features for all points in the dataset
3. Create the codebook of visual words using selected clustering method (k-means or SOM) and given vocabulary size
4. Select only relevant words – those which appear on trained signs
5. Calculate and store the fingerprints of signs – angle and length are stored for each pair of features in a sign

6.2 Detection

1. Detect all key points in the image, assign visual words based on codebook

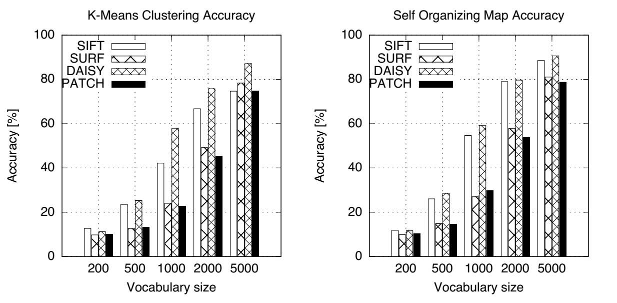

Figure 6: The accuracy of sign recognition using evaluated descriptor types and vocabulary sizes using k-means clustering and self organizing map. Self organizing map yields on average 4% better results.

2. Evaluate all combination of two words in the image:
   (a) Ignore word pairs which do not appear in the trained sign detector
   (b) Look up the stored angle and length for the given pair
   (c) Calculate relative angle and scale
   (d) Remove all pairs which are above the threshold of maximum rotation and scale difference
   (e) Plot a contribution of rotational/scale difference of the remaining points in Hough space
   (f) Find a local maximum in the Hough space above a threshold
   (g) Use second Hough space to vote for position of the sign in the image
   (h) Sign is found if the number of selected pairs is above a threshold

7. Results

The experiments were conducted on a desktop PC with Intel Q6600 2.4 GHz 4 core processor, and a NVIDIA GeForce GTX 460, which was used for clustering. Our sign recognition algorithm is performed on CPU.

Our detection algorithm was run for evaluated descriptors and varying vocabulary sizes. The evaluation was run on 175 images for all 11 signs, and a boolean value was recorded, determining whether a sign was identified in the image. An image may contain multiple signs, so the detector was run independently for each sign on each image.

The accuracy (rate of true positives and true negatives) of our method depending on the clustering method, descriptor type and vocabulary size is shown in Fig 6. Clearly bigger vocabulary size leads to better accuracy. Out of all descriptor types, the Daisy descriptor gives the best results, with accuracy reaching 90% using the SOM. The SOM show consistently better results than k-means clustering, with the difference being on average 4% at 5000 visual words. The receiver operating characteristics for both clustering methods are shown in Fig 7. In comparison to k-means, SOM leads to lower rate of false positives, especially for the SIFT descriptor. The timing

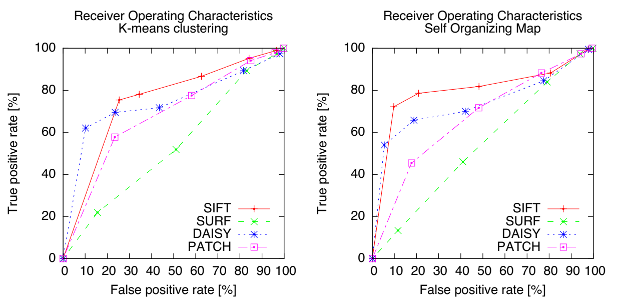

Figure 7: The receiver operating characteristics of evaluated clustering methods.

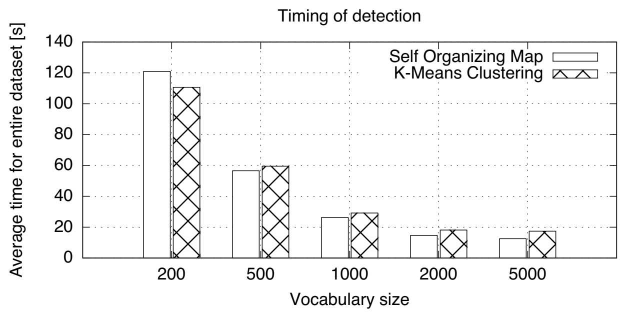

Figure 8: The time required to run the detection algorithm on the entire dataset of 175 images and 11 signs. The time is averaged for all descriptors of a given vocabulary size.

of our method is shown in Fig. 8. The time represents the average time of the detection using the given clustering method in seconds over the entire dataset. As was stated earlier, the more specific a visual vocabulary is, the better it can rule out irrelevant pairs early in the detection process and discard a detection of a sign on an image. Except for vocabulary size of 200 words, SOM shows better performance than k-means algorithm. Using 5000 word SOM vocabulary, the average time of detection of a sign on one image vocabulary is 6.4 ms. This time represents only the time of the detection, not including the time required to identify visual words in an image.

8. Conclusion

We have compared the accuracy and speed of our sign recognition algorithm [3] using varying vocabulary sizes, local feature descriptors and vocabulary construction methods. The self organizing map construction leads to slightly better accuracy and run time, which could be caused by better preserving the topology of the feature space compared to k-means clustering. Out of evaluated descriptors, the Daisy descriptor show the best accuracy. Without taking into account the time required to compute the feature descriptors and identify visual words in an image, our sign recognition algorithm is very fast with possibility of a real-time application.

References

[1] D. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[3] W. Benesova, M. Kottman, and O. Sidla. Hazardous sign detection for safety applications in traffic monitoring. In *Proceeding of SPIE 8301*, 2012.

[4] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *European Conference on Computer Vision*, 2006.

[5] T. Kohonen. Neurocomputing: foundations of research. chapter Self-organized formation of topologically correct feature maps, pages 509–521. MIT Press, Cambridge, MA, USA, 1988.

[6] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[7] D. G. Lowe. Object recognition from local scale-invariant features. *Computer Vision, IEEE International Conference on*, 2:1150, 1999.

[8] D. D. Mauá. Visual topics: The latent dirichlet allocation model in computer vision. Technical report, Escola Politécnica, USP, 2009.

[9] K. Stoffel. Parallel k/h-means clustering for large data sets. *Euro-Par'99 Parallel Processing*, pages 1451–1454, 1999.

[10] S. Taylor and T. Drummond. Binary histogrammed intensity patches for efficient and robust matching. *Int. J. Comput. Vision*, 94:241–265, September 2011.

[11] E. Tola, V. Lepetit, and P. Fua. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. volume 32, pages 815–830, May 2010.

[12] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, MIR '07, pages 197–206, New York, NY, USA, 2007. ACM.