Real-time Visual Concept Classification

J.R.R. Uijlings, A.W.M. Smeulders and R.J.H. Scha

Abstract—**As datasets grow increasingly large in content based image and video retrieval, computational efficiency of concept classification is important. This paper reviews techniques to accelerate concept classification, where we show the trade-off between computational efficiency and accuracy. As a basis we use the Bag-of-Words algorithm that in the 2008 benchmarks of TRECVID and PASCAL lead to the best performance scores. We divide the evaluation in three steps: (1) Descriptor Extraction, where we evaluate SIFT, SURF, DAISY, and Semantic Textons. (2) Visual Word Assignment, where we compare a k-means visual vocabulary with a Random Forest and evaluate subsampling, dimension reduction with PCA, and division strategies of the Spatial Pyramid. (3) Classification, where we evaluate the χ^2, RBF, and Fast Histogram Intersection kernel for the SVM. Apart from the evaluation, we accelerate the calculation of densely sampled SIFT and SURF, accelerate nearest neighbour assignment, and improve accuracy of the Histogram Intersection kernel. We conclude by discussing whether further acceleration of the Bag-of-Words pipeline is possible.**

Our results lead to a 7-fold speed increase without accuracy loss, and a 70-fold speed increase with 3% accuracy loss. The latter system does classification in real-time, which opens up new applications for automatic concept classification. For example, this system permits 5 standard desktop PCs to automatically tag for 20 classes all images that are currently uploaded to Flickr.

I. Introduction

Over the last decade there has been an explosive growth of available multimedia on the internet. Good examples are the photo sharing website Flickr, hosting billions of images with thousands of uploads each minute, and the video sharing website YouTube, hosting millions of videos with hours of video uploaded each minute. The advent of such collections has sparked research on image and video retrieval within these collections, see Snoek and Worring for a recent overview [1]. One active line of research is on using only the visual contents for concept-based search tasks.

Within this domain the Bag-of-Words method [2], [3] has proven to be the most efficient strategy as a generic classification scheme for individual concepts. This is proven by their top performance in various major benchmarks over the past few years such as the TRECVID high-level feature extraction task (which uses video) [4] and the Pascal VOC Challenge (which uses images) [5]. In these benchmarks, concept detectors are able to detect classes such as *chair*, *cat*, *car*, *boat*, *building*, *meeting*, and *sports* with varying degrees of success [4], [5].

But while Bag of Words systems give superior classification results, they are computationally very expensive. For example, a complete classification run for the TRECVID high-level feature extraction task requires processing over 40,000 video frames (1 frame per shot of 180 hours of video). The state-of-the-art system used in [6] will take days to complete on a computer cluster. Hence, for large scale image retrieval with Bag-of-Words there is the need to speed up this method.

This paper presents a comprehensive evaluation of various fast Bag-of-Words components in terms of both computational efficiency and retrieval performance, which is our main contribution. The evaluation shows an increase of accuracy for Random Forests [7] by combining them with Principal Component Analysis, and an increase in computational efficiency by determining relevant image divisions for the Spatial Pyramid [8]. Additionally, we present several improvements upon the components under consideration: We provide a modified way to speed up the calculation of densely sampled SIFT [9] descriptors. Similarly, we turn SURF [10] into a faster, densely sampled descriptor. We accelerate nearest neighbour assignment. Furthermore, we increase accuracy of the Histogram Intersection based Support Vector Machine by balancing visual word frequencies. Finally, next to the experimental evaluation, we provide a theoretical discussion on computational efficiency of the Bag-of-Words method.

This paper is an extension of [11]. This document is structured as follows: First we give a short overview of the Bag-of-Words framework in section II. Section III discusses related work. In section IV we discuss various strategies for accelerating the Bag-of-Words pipeline. Section V gives an overview of our experimental setup. Section VI presents and discusses our results. A discussion about the theoretical computational complexity of the Bag-of-Words pipeline is given in section VII. Finally our conclusions are given in section VIII.

II. Bag-of-Words

The Bag-of-Words method is derived from text retrieval where documents are represented as the frequencies of their words. This word frequency count is used for retrieving or classifying documents. Similarly, in the Bag-of-Words method one first samples local regions from an image which are then converted to *visual* words. The visual word frequencies of an image are then used in subsequent classification. Figure 1 shows a schematic overview of this process, with the creation of a vocabulary of visual words on the left side, and conversion of an image to a visual word frequency histogram on the right.

The visual vocabulary within a Bag-of-Words framework is learned from a training set. First, small image regions are sampled from all images in the training set. From each region a descriptor is extracted, usually SIFT [9] which is a histogram of oriented gradients. Typically, the visual vocabulary is learned by using an unsupervised clustering algorithm such as k-means. The resulting cluster centres define the visual vocabulary in a nearest neighbour sense by partitioning the descriptor space. Each resulting partition represents a visual word. A fast alternative method evaluated in this paper is to partition the descriptor space using binary decision trees.

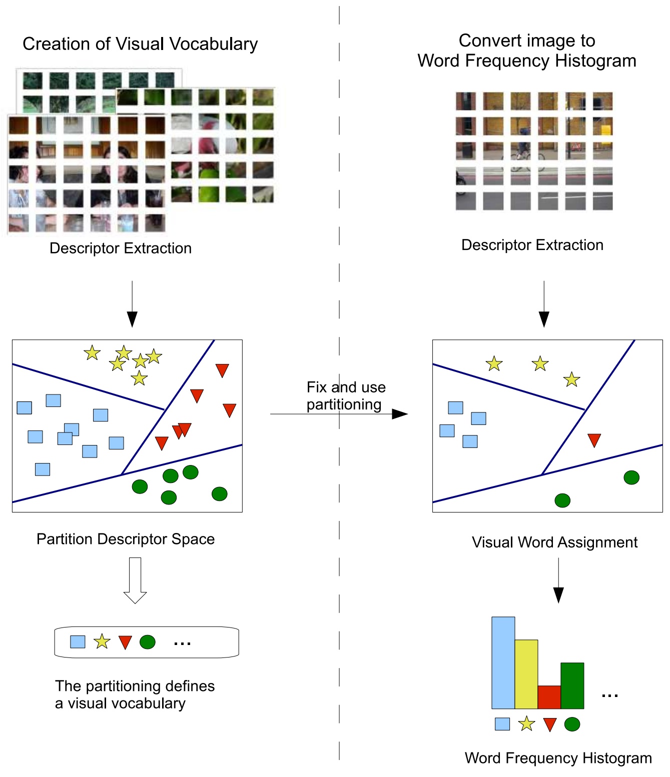

Creation of Visual Vocabulary
Convert image to Word Frequency Histogram
Descriptor Extraction
Descriptor Extraction
Fix and use partitioning
Partition Descriptor Space
Visual Word Assignment
The partitioning defines a visual vocabulary
Word Frequency Histogram

Fig. 1. A schematic overview of creating visual word frequency histograms in the Bag of Words method. For more details see section V.

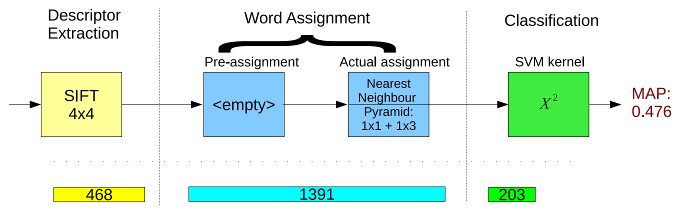


Fig. 2. An example scheme for classification using the Bag of Words method used by good image/video retrieval systems [6], [12], [13]. This particular pipeline takes 2063 ms per image and has an accuracy of 0.476 MAP. State of the art results can be obtained by using dense sampling on multiple scales and by combining a variety of (colour) descriptors in the classification phase.

The size of the visual vocabulary is user defined but typically ranges in the thousands. The visual vocabulary is learned once and remains fixed afterwards.

To create a visual word frequency histogram, small regions are selected from an image from which descriptors are calculated. Each descriptor is then mapped to a visual word of the vocabulary according to the partitioning of the descriptor space. This results in a visual word frequency histogram which is used in subsequent classification.

This paper presents an evaluation of the on-line parts of a Bag-of-Words image classification system. This excludes the creation of the visual vocabulary and learning the classifier. A typical Bag-of-Words image classification pipeline which forms the basis of several state-of-the-art image/video retrieval systems [6], [12], [13] is given in figure 2. The pipeline can be decomposed into three different components: *Descriptor Extraction*, *Word Assignment*, and *Classification*. In this pipeline, the Descriptor Extraction phase extracts SIFT descriptors [9]. The visual vocabulary is created using k-means and Word Assignment is done using nearest neighbour assignment. A weak form of spatial information is incorporated using the spatial pyramid [8]. Classification is done using a Support Vector Machine with a χ^2-kernel.

III. Related Work

We divide the related work into the three components of the Bag-of-Words pipeline: Descriptor Extraction, Word Assignment, and Classification.

A. *Descriptor Extraction*

Descriptor extraction begins with selecting small regions from the image. Lowe [9] introduced a region or interest point detector based on the difference of Gaussian (DOG). Bay *et al.* [10] improved computational efficiency with a factor six by proposing a Fast-Hessian interest point detector. Šochman and Matas [14] accelerated interest point detection by emulating them using their WaldBoost algorithm. However, in the context of object recognition, Jurie and Triggs [15] showed that sampling many patches on a regular dense grid outperforms the use of interest points. This paper therefore uses the dense sampling method which eliminates the computational costs of detecting interest points altogether.

From the small, selected regions in the image one extracts descriptors. Mikolajczyk and Schmid [16] compared various descriptors and found SIFT [9] or SIFT-like descriptors to be the best in the context of image matching under various image transformations. Mikolajczyk *et al.* [17] and Zhang *et al.* [18] showed that SIFT-like descriptors are also superior for object detection. In this paper we introduce a computationally efficient SIFT-variant which gives similar accuracy to the original SIFT.

The SIFT descriptor consists of oriented gradient responses that are summed over subregions. The summation becomes computationally expensive when lots of descriptors are extracted. Grabner *et al.* [19] proposed to obtain the summation efficiently by using integral images enabling them to calculate the summation of any subregion using only its four corners. The resulting descriptor deviates from SIFT [9] in omitting two weighting schemes: First of all, SIFT uses a linear interpolation between subregions which makes it robust against small changes in position. Second, SIFT has a Gaussian weighting scheme around the origin emphasizing the importance of the detected interest point. [19] report a factor eight speed increase while matching performance decreases slightly. Unlike [19], we do not use interest points and we exploit the regularity of the dense sampling to create fast summations over subregions.

Tola *et al.* [20] propose a different method to speed up the summations over subregions. They use a Gaussian convolution on the pixel responses after which each pixel represents a weighted sum of responses over its neighbourhood. They then create their DAISY descriptor by taking circularly arranged responses. As [20] can be seen as a form of dense sampling, we include DAISY in our evaluation.

SIFT is based on relatively expensive oriented gradient responses. This is addressed by Bay *et al.* [10] with SURF, a spatial descriptor similar to SIFT based on Haar wavelet responses. Haar wavelets are cheaper to compute than the Gaussian derivatives of SIFT. We include the SURF descriptor in our experiments and examine if their results on object recognition extend to larger and more difficult datasets. Additionally, as with SIFT, we exploit the spatial nature of SURF in combination with the dense sampling strategy to speed up calculation.

Shotton *et al.* [21] propose to omit the extraction of descriptors altogether in the context of pixel-wise classification and segmentation. Instead, they use raw-pixel values of a small region directly. For the subsequent step of Word Assignment, they use a Random Forest, a set of supervised decision trees. We include their Semantic Textons in our evaluation.

B. *Visual Word Assignment*

Descriptors are assigned to a visual vocabulary by what we call Word Assignment. Commonly, large visual vocabularies are created with unsupervised k-means clustering which gives good performance (e.g. [12], [18], [22]). Typically each descriptor is assigned to a single word of this visual vocabulary using nearest neighbour *e.g.* [18], [22], [23]. However, it has been shown that assigning each descriptor to multiple visual words by using a so-called soft-assignment is beneficial to performance [24], [25] at the expense of extra calculation time: In soft assignment on needs to obtain the closest n visual words and calculate a posterior probability over these. As this paper aims for acceleration of visual concept classification we exclude the use of soft-assignment in our evaluation.

Computation time for the Word Assignment is dependent on three factors: the number of descriptors extracted per image, the dimensionality of the descriptors, and the size of the visual vocabulary. Nowak *et al.* [26] showed that using more descriptors is better. We evaluate the speed-performance trade-off for using fewer descriptors. Mikolajczyk and Schmid [16] apply Principal Component Analysis (PCA) on their SIFT-variant called GLOH to ensure it has the same dimensionality as SIFT. We apply PCA to our descriptors to reduce their dimensionality in order to increase computation speed.

Jiang *et al.* [27] and Moosmann *et al.* [7] experimented with the size of the visual vocabulary and concluded that large vocabularies are generally beneficial for large datasets. We will fix the size of the visual vocabulary to 4096 visual words as this gives good results on both the TRECVID and Pascal VOC dataset [6], [12], [13]. Jurie and Triggs [15] and Wang *et al.* [28] propose methods to create compact yet discriminative vocabularies. These methods are orthogonal to the methods described in this paper.

In order to make word assignment more efficient, several tree-based assignment algorithms were proposed [29]–[31]. Tree-based word assignment algorithms allow for a logarithmic rather than a linear assignment time in terms of the number of visual words. The most interesting one is the work on supervised random forests by Moosmann *et al.* [30]; apart from a computational advantage the paper also reports improvements over regular k-means in terms of performance on their four-class datasets. As the creation of the trees is supervised, the relatively small number of classes they use might have positively influenced their results. This paper considers how their method extends to more classes.

As mentioned earlier, Shotton *et al.* [21] use Random Forests directly on the pixel values of image regions and show its effectiveness in image segmentation. Their Random Forests differ from [30] in that each decision node in the tree works on multiple values of the descriptor instead of one. In most of our experiments we will use the Random Forests which work on a single dimension as these are faster and preliminary experiments showed no significant benefits for decisions on multiple dimensions. Only when evaluating their Semantic Textons we use decision trees as defined in [21].

Lazebnik *et al.* [8] proposed the spatial pyramid, introducing a weak form of spatial information by increasingly subdividing the image and obtain a visual word frequency histogram for each region separately. This results in a 5-10% performance increase at a very limited computational *word assignment* cost. He *et al.* [32] expanded on this work by learning weights for each of the resulting subregions. However, classification time is dependent on the size of the word frequency histogram and hence on the number of subregions. In this paper we consider various image division strategies to minimize the number of subregions and speed up classification time.

C. *Classification*

Support Vector Machines (SVMs) are a very popular classifier due to its robustness against large feature vectors and sparse data. They are successfully used in Bag of Words methods. The choice of SVM-kernel has a large impact on performance. Both Zhang *et al.* [18] and Jiang et al. [27] determined that the χ^2-kernel gives the best accuracy. We will follow their experiments but next to retrieval performance we also focus on computational efficiency.

Maji *et al.* [33] proposed an efficient classification scheme for SVMs when using histogram intersection kernels. In this paper we will use their implementation for the histogram intersection kernel.

D. *Graphics Processing Unit*

Orthogonal to methodological improvements, researchers are looking to Graphics Processing Units (GPUs) to speed up Bag-of-Words . While hardware optimizations fall outside the scope of this paper, the methods we evaluate in this paper can all be implemented on a GPU: To calculate SIFT and SURF one needs matrix multiplications and (recursive) Gaussian filters, which can be found in the standard GPU software development kit CUDA [34]. Sharp [35] proposes an algorithm for Random Forests on the GPU. Finally, Catanzaro *et al.* [36] show how to implement a Support Vector Machine on the GPU.

IV. Accelerated Bag-of-Words

In this section we describe the simple way of calculating densely sampled descriptors. Furthermore we discuss a fast way to calculate nearest neighbour assignment. Then we will discuss the Random Forests of Moosmann *et al.* [7] and the fast Histogram Intersection SVM classifier of Maji *et al.* [33].

A. Descriptor Extraction

1) Accelerated Dense Descriptor Extraction: Both SIFT and SURF are spatial descriptors: each is constructed of 4×4 subregions which in turn are described by the summation of pixel-wise responses over an area. In the case of SIFT the responses are oriented gradients calculated using image convolutions, for SURF these are Haar wavelet responses calculated using simple summations and subtractions.

First we observe that if the dense sampling rate is the same as the size of a subregion we can reuse these subregions for the other descriptors. For the original 4×4 SURF and SIFT descriptors this means a factor 16 speed improvement for the summations over the pixel responses.

The original SIFT uses a Gaussian weighting over the complete image patch, attributing greater importance to the values in the middle of the descriptor. This step obstructs the reuse of subregions. But the original SIFT was made for use in conjunction with interest points where the middle of the descriptor is more important by design. However, the dense sampling strategy creates arbitrary image patches hence all parts of an image patch seem equally important. As we use dense sampling, we omit the Gaussian weighting.

To sum the responses within each subregion we use two matrix multiplications: One to sum in the row direction and the other to sum over the column direction. Consider the pixel-wise responses R from an image. If we want to sum the responses over subregions of 3 by 3 pixels we employ a matrix multiplication ARB, where A sums over elements in the row direction and has the form of

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 \end{pmatrix}.$$

Matrix B sums over the column direction and is similar to A^T but has a size adapted to R.

For robustness against small shifts in position of the descriptor, SIFT uses a linear weighting to divide responses over neighbouring subregions. We do a linear weighting by modifying A (and likewise B) to

$$\begin{pmatrix} 1 & 1 & 2/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1/3 & 2/3 & 1 & 2/3 & 1/3 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 2/3 & 1 & 2/3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where the top left entry is 1 rather than $1/3$ due to normalization at the boundary. The resulting descriptor only differs from the original SIFT in omitting the Gaussian weighting over the whole descriptor. It differs from [19] in that they omit also the linear interpolation between subregions while we keep this.

Performing the summation over the regions with matrix multiplications allows the use of specialized matrix multiplication libraries in combination with sparse matrices. The resulting speed-up is up to a factor 2 compared to a naive C++ implementation.

Once all values of subregions have been extracted we can construct any spatial form of these subregions. Traditionally the SIFT (and SURF) descriptor is composed of 4×4 subregions. In this paper we also evaluate 2×2 descriptors which have less dimensions.

B. Visual Word Assignment

1) Nearest Neighbour Assignment: Each descriptor from an image is projected onto a predefined visual vocabulary (obtained by k-means) using nearest neighbour assignment. It is common practice to normalize both SIFT and SURF to unit vectors to make them robust against illumination changes. As all descriptors are unit length, nearest neighbour assignment using the Euclidean Distances is equal to using angles between vectors. By using the inner product to calculate these angles, we obtain a speed-up of 43% over using Euclidean Distances in the word assignment phase.

2) Random Forest: A random forest is a collection of binary decision trees whose combination leads to fast yet accurate classification performance. This paper evaluates the decision trees in the word assignment step. Unlike k-means visual vocabularies, the decision trees are created in a *supervised* way. Following Moosmann *et al.* [7], we use [37] for the construction of the trees. For each tree we start with 250,000 labelled descriptors D from our training set, where the labels are taken from the global image annotations (*i.e.* annotations at image level). Learning is done recursively. At each node n, s random splits are proposed by choosing a random dimension of the descriptors and a random threshold t. This splits the set of descriptors D_n at node n in D_a and D_b. Each split is evaluated using the information gain ΔE, defined as [21]

$$\Delta E = -\frac{|D_a|}{|D_n|}E(D_a) - \frac{|D_b|}{|D_n|}E(D_b), \tag{1}$$

where $E(D_a)$ is the Shannon Entropy of the class labels of D_a. The split with the highest information gain is then adopted. Training continues with D_a and D_b and stops if a specific depth is reached. The end result is a binary decision tree.

Word assignment using a random forest is done by obtaining the visual word frequency histograms for all trees and simply concatenating these histograms into one vector.

Random forests are interesting from a computational point of view in two ways: First of all, the binary nature of the decision trees result in a word assignment time which is logarithmic in the number of visual words, whereas this is linear for nearest neighbour assignment. Furthermore, as at each decision node in the tree only a single dimension of the descriptor is compared to a threshold, the visual word assignment time for random forests is independent of the dimensionality of the descriptors while for nearest neighbour assignment this dependency is linear.

C. Classification

1) The precomputed kernel: The classification function for a SVM can be written as [38]

$$h(\mathbf{x}) = b + \sum_{j=1}^{m} \alpha_j t_j K(\mathbf{x}, \mathbf{z}_j), \tag{2}$$

where $\mathbf{x} = \{x_1, \ldots, x_k\}$ is the vector to be classified, $\mathbf{z}_j = \{z_{1j}, \ldots, z_{kj}\}$ is the j-th support vector, α_j is its positive support vector weight, $t_j \in \{+1, -1\}$ is the label of the support vector, m is the number of support vectors, and $K(\cdot, \cdot)$ is a kernel function. The time complexity of this function is dependent on its kernel function. For the common distance-based kernel functions that we use the time complexity is $\mathcal{O}(km)$. Suppose that the average number of support vectors for a single class is $\bar{m}$, then classification for all classes is of complexity $\mathcal{O}(k\bar{m}c)$.

Alternatively, as support vectors are selected from the training set one can choose to calculate the kernel function between the whole training set and a test sample and select later for each class which kernel entries to use. If we define q as the size of the training set, classification complexity for all classes becomes $\mathcal{O}(kq)$. This strategy is computationally advantageous if $q < \bar{m}c$. For our datasets this is indeed the case: all datasets have 15 or more classes and each class takes 10-25% of the training vectors as support vectors.

In our experiments we calculate the kernel function between the whole training set and the whole test set at once, which is called using a pre-computed kernel. Compared to calculating the distance separately for each test sample, the pre-computed kernel reduces overhead and may allow a speed-up when using efficient matrix multiplications but its memory requirements are much higher.

2) Efficient Support Vector Machine classification for Histogram Intersection kernels: The classification function for a Support Vector Machine in equation 2 has time complexity $\mathcal{O}(km)$. Recently, Maji *et al.* [33] showed that the classification function for the histogram intersection kernel

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{k} \min(x_i, z_i) \tag{3}$$

can be rewritten to give a time complexity of $\mathcal{O}(k \log m)$:

$$\begin{aligned} h(\mathbf{x}) &= b + \sum_{j=1}^{m} \alpha_j t_j K(\mathbf{x}, \mathbf{z}_j) & (4) \\ &= b + \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_j t_j \min(x_i, z_{ij}) & (5) \end{aligned}$$

Now for a fixed value of i let $\tilde{z}_{ij}$ denote the sorted values of z_{ij} in ascending order with corresponding weights $\tilde{\alpha}_j$ and labels $\tilde{t}_j$. Let r be the largest integer for which $\tilde{z}_{ir} \leq x_i$. This allows rewriting equation 5 as

$$h(\mathbf{x}) = b + \sum_{i=1}^{k} \left\{ \sum_{j=1}^{r} \tilde{\alpha}_j \tilde{t}_j \tilde{z}_{ir} + x_i \sum_{j=r+1}^{m} \tilde{\alpha}_j \tilde{t}_j \right\}. \tag{6}$$

For each i we can precompute for all r

$$\sum_{j=1}^{r} \tilde{\alpha}_j \tilde{t}_j \tilde{z}_{ir} \tag{7}$$

and

$$\sum_{j=r+1}^{m} \tilde{\alpha}_j \tilde{t}_j \tag{8}$$

as they have become independent of x_i. By exploiting the piecewise linear form of the resulting decision function the memory requirement becomes twice that of the normal SVM implementation [33]. Calculating the classifier output now amounts to finding for each dimension i the correct r which can be done logarithmically in the number of support vectors m. The total run-time complexity thus becomes $\mathcal{O}(k \log m)$ instead of $\mathcal{O}(km)$, which is a significant speed increase.

Maji *et al.* [33] also found that the piece-wise linear function between the brackets of equation 6 can reasonably well be approximated by a piece-wise linear function with uniform spacing between its segments (rather than a spacing which is determined by the sorted values $\tilde{z}_{ij}$ of the support vectors). The uniform spacing allows for a direct mapping of x_i to a corresponding r, removing the dependence on the number of support vectors m altogether. This leads to an even faster classification complexity of order $\mathcal{O}(k)$ with negligible loss of accuracy. We evaluate both the exact and approximate Histogram Intersection SVM of [33].

V. Experimental Setup

We compare various alternative Bag-of-Words components with respect to a basic Bag-of-Words pipeline. In our comparison we consider both retrieval performance and computational efficiency. We divide our experiments into the three stages of the Bag-of-Words pipeline:

I **Descriptors.** The features which describe the extracted local image patches.

II **Word Assignment.** The assignment of these descriptors to a word in the visual vocabulary, resulting in a visual word frequency histogram. The Spatial Pyramid is applied in the word assignment phase.

III **Classification.** The classification of these visual word frequency histograms.

For measuring retrieval performance we report the standard measure for that dataset. This is either the percentage of correctly classified examples or the Mean Average Precision (MAP) over all classes. The Average Precision for a single class is defined as

$$\frac{1}{m} \sum_{i=1}^{n} \frac{f_c(x_i)}{i}, \tag{9}$$

where: n is the number of images. m is the number of images of class c. x_i is the i-th image in the ranked list $X = \{x_1, \cdots, x_n\}$. Finally, f_c is a function which returns the number of images of class c in the first i images if x_i is of class c, and 0 otherwise.

Computational efficiency is measured in milliseconds per image, where the measurement is an average over all images in the test set. Classification time for an image is reported for the classification of all classes in the dataset. The efficiency measurements are done on a mainstream processor (a single core of a 3.16 Ghz Intel Core Duo E8500 processor).

A. Datasets

We perform our experiments on three different datasets. The **Pascal VOC 2007** dataset consists of 9963 images divided into a predefined training and test set of respectively 5011 and 4952 images. The general image size is 300×500 pixels. The dataset consists of 20 object classes: *aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining-table, dog, horse, motorbike, person, potted-plant, sheep, sofa, train*, and *TV/monitor*. Some images contain multiple objects of potentially different classes. Retrieval performance is measured using the Mean Average Precision (MAP) over all object classes.

The **MediaMill Challenge** dataset [39] consists of 85 hours of video from the TRECVID 2005 development set which is segmented into shots with corresponding keyframes. As is common, we only use these keyframes, discarding temporal information (unlike *e.g.* [40]). The set is divided into a predefined training and test set which respectively contain 30993 and 12914 keyframes. All keyframes are 240×352 pixels and contain compression artefacts. There are 101 concept categories ranging from object categories (*e.g. bird*, *bus*), to natural scenes (*e.g. mountain*, *sky*), semantic concepts (*e.g. entertainment*, *meeting*), and actions (*e.g. fire weapon*, *people marching*). Keyframes may contain zero, one, or multiple concepts.

The **Fifteen Scene Categories** dataset [8] expands upon earlier scene category databases [41], [42]. For each scene category there are 200-400 images with an average size of 300×250 pixels. We randomly divide the set into a training and test set of respectively 2245 and 2240 images. The scene categories for this dataset are mutually exclusive and contain categories like *bedroom*, *industrial*, and *forest*.

In our experiments we measured very similar speed improvements for all three datasets and also similar trends in accuracy. Therefore we will discuss only the results for the Pascal VOC 2007 dataset in detail. We will return to the other datasets in the conclusions.

B. Baseline

Our baseline Bag-of-Words system is modelled after the best systems of the Pascal VOC challenge 2007 and 2008 [12], [13], where we exclude the spatial pyramid for its computational demands during training. However, as the spatial pyramid seems intuitively equally powerful for all descriptors and word assignment methods we do not have to include it for most of our experiments.

We use the intensity-based SIFT descriptor extracted by our fast Dense Sampling strategy, termed DIFT from now on. We sample subregions of 6 by 6 pixels each 6-th pixel and use the original spatial configuration of 4×4 subregions.

Our visual vocabulary consists of 4096 words created using k-means clustering. This vocabulary size is kept constant throughout our experiments. New descriptors are projected to the visual vocabulary using nearest neighbour assignment through inner products. Classification of the resulting word frequency histograms is done using a SVM with a χ^2-kernel, where we make use of the precomputed kernel which takes up most of the classification time.

The resulting Bag-of-Words pipeline for the baseline experiment is presented in figure 3. Note that the pre-assignment step is currently empty but will be used by two of our experiments.

Subsequent experiments will always affect a single element of this baseline pipeline.

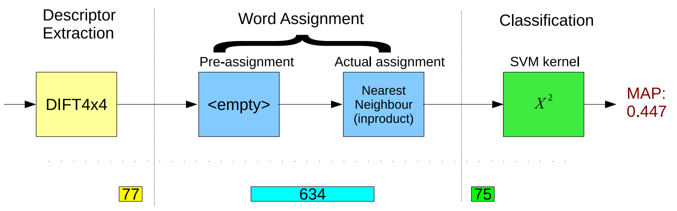


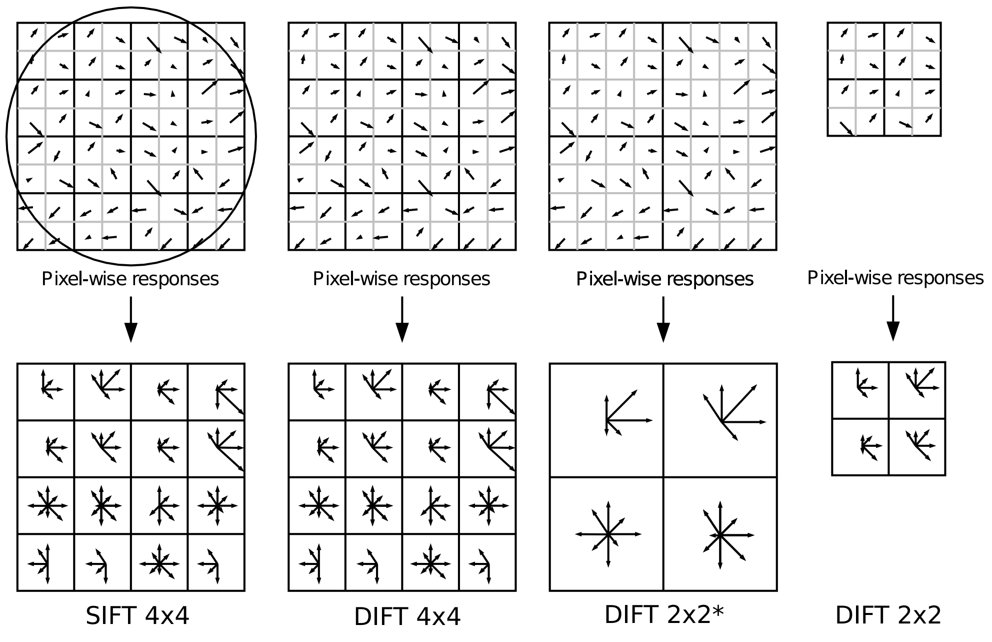
Fig. 3. The baseline Bag of Words pipeline as used in our experiments. The total computation time is 786 milliseconds per image to classify all 20 classes of the Pascal VOC dataset. Its Mean Average Precision is 0.447.

C. Descriptors

In this experiment we compare various fast alternatives to the SIFT descriptor, all extracted using dense sampling on a regular grid. We focus on computational efficiency for both the extraction of the descriptors and the visual word assignment because the dimensionality of the descriptors influences assignment time.

We compare four SIFT variants and three SURF variants. We compare our implementation of the original SIFT 4×4 descriptor which includes the Gaussian weighting with our DIFT 4×4 descriptor which omits this weighting. Both descriptors have 128 dimensions. To decrease the dimensionality and hence word assignment time we construct two DIFT versions out of four subregions resulting in 32 dimensions. DIFT 2×2 uses the same subregions as DIFT 4×4. DIFT 2×2* uses the exact same pixel values as DIFT 4×4. This arguably results in a more fair comparison but is less compatible with our descriptor extraction method: we need to extract the features four times to achieve the same dense sampling rate (we need four different sets of subregions). Notice that besides a dimensionality reduction, fewer subregions also result in less redundancy: neighbouring subregions are likely to describe similar statistics, so more subregions in a descriptor means a higher probability of redundancy. A schematic overview of the various SIFT variants is given in figure 4.

We also use three variants of DURF which can be seen as the counterparts of DIFT; the responses within the same subregions are used for summation but DURF uses Haar wavelet responses rather then oriented gradients. As the Haar wavelets are calculated in only the horizontal and vertical directions and not in the diagonal directions, the resulting dimensionality of the descriptors is half that of DIFT. The DURF variants are termed DURF 4×4, DURF 2×2, and DURF 2×2*, and consist of respectively 64, 16, and 16, dimensions.



(a) (b) (c) (d)

Fig. 4. A schematic overview of the various SIFT variants used in this paper. The circle in figure 4(a) represents a Gaussian weighting centred around the origin of the descriptor. In this picture most subregions are 2 by 2 pixels while in our actual experiments these subregions are 6 by 6 pixels.

Descriptor	Region size	#dimensions
SIFT 4×4	24 by 24	128
DIFT 4×4	24 by 24	128
DIFT 2×2*	24 by 24	32
DIFT 2×2	12 by 12	32
DAISY	24 by 24	200

Descriptor	Region size	#dimensions
-	-	-
DURF 4×4	24 by 24	64
DURF 2×2*	24 by 24	16
DURF 2×2	12 by 12	16
Textons	24 by 24	576

TABLE I
REGION SIZE AND DIMENSIONALITY OF THE VARIOUS DESCRIPTORS USED.

We also evaluate the DAISY descriptor [20]. The DAISY descriptor can be seen as a variant of SIFT, where its subregions are circular and summed using a Gaussian weighting rather than square and summed using a linear weighting.This is visualised in figure 5. In total there are 25 subregions of 8 oriented gradients, resulting in a 200 dimensional vector. We use the implementation provided by [20].

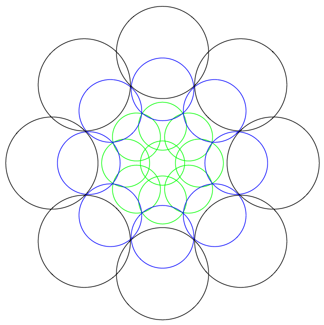


Fig. 5. Visualisation of pixel-wise summation as done by the DAISY descriptor. Each circle describes a Gaussian summation over the pixel-wise oriented gradient responses.

Finally, we evaluate Semantic Textons [21]. Semantic Textons are defined to work with a Random Forest and are not suited for nearest neighbour assignment. For fair comparison, we use Semantic Textons on the intensity values only, just like the other descriptors. A concise overview of the dimensionality and region sizes of the descriptors is given in table I.

D. Word Assignment

The word assignment time when using nearest neighbour assignment depends on three factors: the size of the visual vocabulary, the number of descriptors generated per image, and the dimensionality of the descriptors. In this paper we discuss experiments on the number and dimensionality of the descriptors. We refrain from experimenting with the size of the visual vocabulary, as this has been exhaustively studied in [7], [27]. Instead we fix the vocabulary size to 4096, which gives good results on the Pascal VOC dataset [12], [13]. Using a larger vocabulary implies an increased classification time for the SVM.

We decrease the number of descriptors per image by a random sub-sampling strategy.

The size of the descriptors is reduced by Principal Component Analysis. We only use the rotation component of PCA. The translation component is expensive to calculate and does not influence distances and hence does not influence the resulting word frequency histograms.

We also consider the Random Forest [7] described in section IV-B2 as a fast alternative to k-means and nearest neighbour assignment. As a Random Forest of 4 trees gives good results for [7], our forests are made of 4 trees of depth 10 resulting in the appropriate vocabulary size of 4096 visual words.

Finally we consider various image divisions for the spatial pyramid of Lazebnik *et al.* [8]. The original spatial pyramid uses the same number of horizontal and vertical divisions. We consider combinations of 1-4 horizontal and 1-4 vertical divisions. The aim is to find the minimum number of resulting sub-regions with high retrieval efficiency, as classification time is linear in the number of these sub-regions.

E. Classification

The choice for Support Vector Machine kernels influences both retrieval performance and classification speed. For a word frequency histogram $\mathbf{x} = \{x_1, \ldots, x_k\}$ and a support vector $\mathbf{z} = \{z_1, \ldots, z_k\}$, the SVM kernel is defined as $K(\mathbf{x}, \mathbf{z})$. In our experiments we compare three different kernels:

1) Histogram intersection kernel, defined as

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{k} \min(x_i, z_i). \tag{10}$$

2) Radial Basis Function, defined as

$$K(\mathbf{x}, \mathbf{z}) = e^{-\gamma \mathbf{D}_{\text{euclid}}}, \tag{11}$$

where

$$\mathbf{D}_{\text{euclid}} = \sum_{i=1}^{k} (x_i - z_i)^2 \tag{12}$$

and γ is a normalization factor which in our experiments we set to $1/\bar{\mathbf{D}}_{\text{euclid}}$ as in [12].

3) χ^2-kernel, defined as

$$K(\mathbf{x}, \mathbf{z}) = e^{-\gamma \mathbf{D}_{\text{chi}}}, \tag{13}$$

where

$$\mathbf{D}_{\text{chi}} = \sum_{i=1}^{k} \frac{(x_i - z_i)^2}{x_i + z_i} \tag{14}$$

and γ is a normalization factor which in our experiments we set to $1/\bar{\mathbf{D}}_{\text{chi}}$ as in [12].

Our implementation for pre-computing these kernels exploits the sparseness of the word frequency histograms for computational efficiency. Besides the pre-computed Histogram Intersection kernel we use the fast implementation of [33].

Finally, we revisit the spatial pyramid using the classification method which proves to be the fastest.

F. Implementation Details

For the DIFT and DURF descriptors we sum responses over subregions of 6 by 6 pixels. The horizontal and vertical responses for SIFT are calculated using a Gaussian derivative filter while the diagonal responses are calculated using a fast anisotropic Gaussian derivative filter [43], all using a sigma of 1. For DURF we calculate each 2nd pixel a Haar Wavelet response of 4 by 4 pixels. Notice that calculating responses each 2nd pixel means that the subregions of 6 by 6 pixels contains only 9 Haar responses instead of the 36 Gradient responses of DIFT. For DAISY we set the radius of the descriptor to 12 pixels resulting in comparable image-regions from which the descriptor is calculated. We use the default settings for the other parameters [20]. For the Semantic Textons we take regions of 24 by 24 pixels where we normalize each region to unit length to be invariant against intensity changes. All descriptors in this paper are sampled at each 6-th pixel. For each descriptor type we generate about 4500 descriptors per image for the Pascal VOC dataset.

One visual vocabulary for nearest neighbour assignment is created using k-means on 250,000 descriptors with $k = 4096$. The word assignment itself is done by using the maximum inproduct as explained in section IV-B1.

We learn the Random Forest using 250,000 descriptors. The number of proposed random splits s had little influence in preliminary experiments. In this paper we set it to half of the number of dimensions of the descriptor. We create 4 trees of depth 10 resulting in 4096 visual words.

For classification we optimize the slack parameter using 3-fold cross validation. We use the prior probabilities to set the weights for the training samples such that the positive and negative training *sets* are weighted equally: Positive examples get weight $1/P(\text{pos})$, negative examples get weight $1/P(\text{neg})$.

Most of our implementation is done using optimized Matlab code. We created C++ implementations (MEX-files) for the random forest assignment, calculating the maximum of a matrix, the χ^2 distance, and the histogram intersection kernel. We used the C++ implementations (MEX-files) of the anisotropic Gaussian Filtering [43], the LIBSVM implementation [44] which allows the use of precomputed kernels, and the adjusted LIBSVM implementation of [33] for the fast histogram intersection.

VI. Results

This section presents all results. For clarity, throughout this section we use percentages to denote computational efficiency and MAP scores to denote retrieval performance.

A. Baseline

The baseline is given in figure 3, using DIFT 4×4, nearest neighbour assignment and a SVM classifier using the χ^2-kernel.

The retrieval effectiveness of this pipeline is 0.447 MAP, comparable with [12], [13] for similar settings. In practice this means that for this dataset, for each class on average 80% of the top ten images and 60% of the top 100 images is correct.

The processing of the total pipeline takes 786 ms, where 10% of the time is used for descriptor extraction, 81% is used for word assignment, and 9% is used for classification. Word Assignment time mainly consists of calculating the inner product between the visual vocabulary and the descriptors. Classification time can be split into 74 ms per image for pre-calculating the χ^2-kernel and 1 ms per image for calculating the classification function for all 20 classes. Notice that the 1 ms of calculating the classification function stays the same throughout all experiments on the Pascal VOC dataset as it is dependent only on the number of training and test samples and on the number of classes.

By sampling at different scales, combining different (colour) variants of SIFT during classification and by using the spatial pyramid we obtain a MAP of 0.57, comparable to the best results reported on this dataset [12], [13].

B. Descriptors

We compare the SIFT, DIFT and DURF descriptors with various spatial configurations as described in section V-C, as well as the DAISY descriptor. Results are given in figure 6.

The retrieval performance among the SIFT 4×4, DIFT 4×4, DURF 4×4, and DAISY descriptors is comparable. Compared to SIFT 4×4, descriptor extraction is 17% faster for Daisy, 500% faster for DIFT 4×4, and 3200% faster for DURF 4×4. Furthermore, compared to both SIFT 4×4 and DIFT 4×4, visual word assignment is 37% faster for DURF 4×4 and 46% slower for DAISY, due to the respectively smaller and larger dimensionality of these descriptors.

The retrieval performance for the 2×2 spatial configurations for DIFT and DURF is respectively 0.01 and 0.02 MAP lower than their 4×4 counterparts. While descriptor extraction speed is only slightly faster, word assignment speed is increased significantly due to the lower dimensionality: For DIFT the speed increase of the 2×2 configuration is 100%, for DURF this is 40%.

The DAISY descriptor is not much faster than SIFT 4×4, contrary to [20]. This is because DAISY is created for sampling at each pixel rather than each 6-th pixel. When sampling each pixel, we measured that DAISY is 2000% faster than SIFT 4×4. However, in such a scenario its speed increase relies on doing summations over subregions using convolutions. This can also be done for both SIFT and SURF, hence DAISY is never faster.

To conclude, DIFT 4×4 and DURF 4×4 have optimal accuracy at a good computation time. DIFT 2×2 and DURF 2×2 are good alternatives when speed is more important than obtaining the highest accuracy. DAISY gives also a good accuracy but is slow relative to DIFT and DURF. Therefore we will not include this descriptor in subsequent experiments.

C. Word Assignment

1) Random Forest: In this experiment we compare the nearest neighbour assignment with Random Forests. We do this on the SIFT, DIFT, and DURF descriptors of our previous experiment. Additionally, we will include the Semantic Textons [21] which are designed to work with Random Forests. Results are shown in figure 7.

Considering the retrieval performance in figure 7(a) we observe that the MAP values for the 4×4 configuration of DIFT stays approximately the same with a MAP decrease of 0.004. DURF 4×4 decreases with 0.026. However, in contrast to nearest neighbour assignment, DIFT 2×2 and DURF 2×2 outperform their 4×4 counterparts. DIFT 2×2 even has a performance comparable to the baseline. Because the 2×2 versions have lower dimensionality and less redundancy, it suggests that Random Forests are sensitive to these aspects. This will be further examined in our PCA experiment in section VI-C3.

The accuracy of the Semantic Textons is comparatively low, with a MAP of 0.364. This result is obtained using a Random Forest which uses only a single dimension in its decision nodes. We also experimented with including decisions on multiple dimensions exactly as in [21]. But both execution time and accuracy were very similar (data not shown). We conclude that Semantic Textons are less powerful than histograms of oriented gradients for image retrieval using Bag-of-Words. We exclude them in subsequent experiments.

In terms of word assignment speed, figure 7(b) shows that the Random Forest does what it is designed for and results in a speed improvement of a 3500-4500%. As it also gives good retrieval performance we include the Random Forest in subsequent experiments.

As discussed in section IV-B2, the number of computational operations performed while doing Random Forest assignment is independent of the dimensionality of the region descriptors. But as can be seen in figure 7(b), higher dimensional descriptors have a larger word assignment time. We attribute this to longer memory access in larger blocks of memory.

2) Subsampling: This experiment explores speeding up visual word assignment by sampling only part of the extracted descriptors from an image. Results are shown in 8.

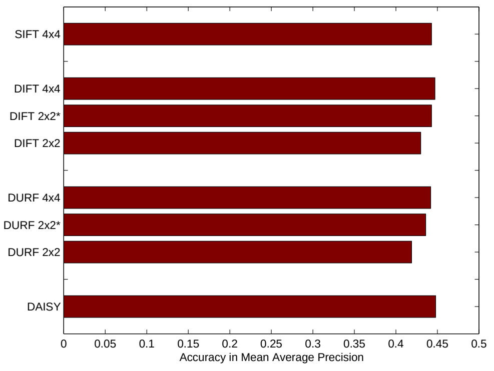
We see that word assignment speed is linear in the number of descriptors. Retrieval performance is bounded: more descriptors are better but it stabilizes at a certain point. An accuracy of 0.314 MAP when using 10% of the descriptors (about 450 descriptors per image) may seem low compared to other published work. Additional tests verified that this is because we use dense sampling rather than interest points (data not shown). As interest points focus on the most important parts of an image and hence generate the most important visual words, fewer descriptors are necessary to get reasonable results.

Using fewer descriptors per image is only viable when speed is more important than accuracy in which case a Random Forest assignment is preferred. Comparing the descriptor extraction time and the Random Forest assignment time we see that the Random Forest is 500% faster, therefore no significant speed-ups can be achieved by subsampling. For DURF 4×4, descriptor extraction time and word assignment time is about equal so here one can gain some extra speed improvement at a considerable loss in accuracy.

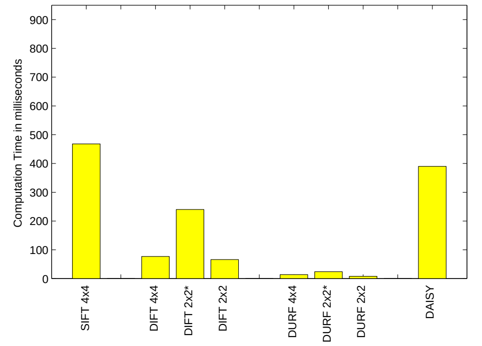
For this dataset we recommend not to use subsampling.

3) PCA: We now use PCA to reduce the dimensionality of the descriptors. Figure 9 shows the results.

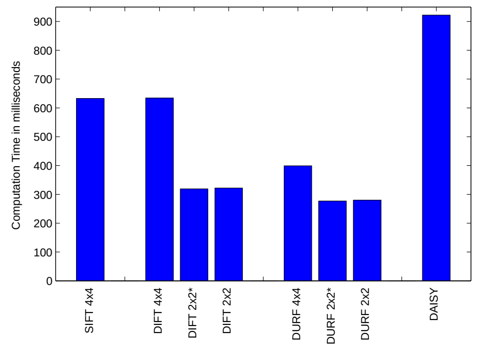
For retrieval effectiveness we observe that the use of PCA



(a) Classification Accuracy

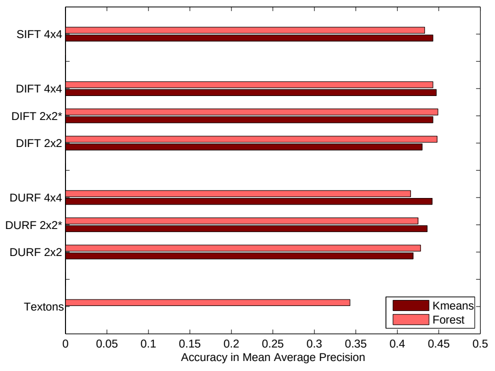


(b) Descriptor Extraction Time

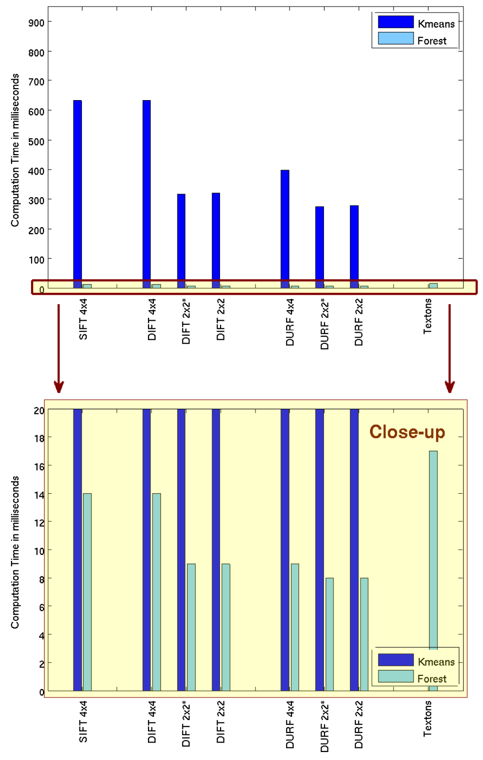


(c) Assignment Time

Fig. 6. Retrieval performance and computation speeds for various descriptors. Speed is for both descriptor extraction and word assignment using nearest neighbour with a k-means visual vocabulary.



(a) Classification Accuracy



(b) Assignment Time

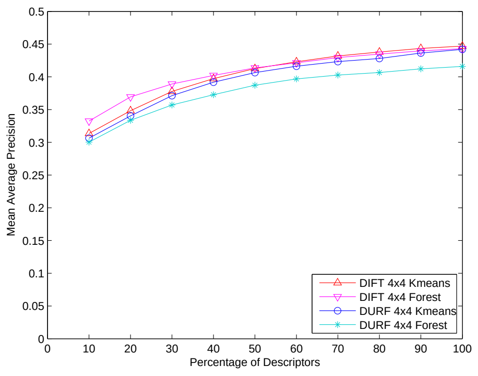
Fig. 7. Random Forests versus k-means nearest neighbour assignment: Retrieval performance and assignment speeds for various descriptors. Textons have no comparison with k-means as they are designed for use with Random Forests only.

increases performance for the Random Forest assignment. For DIFT 4×4 this performance is maximally 0.462, which is 0.015 MAP higher than the baseline and 0.02 MAP higher than the Random Forest without PCA.

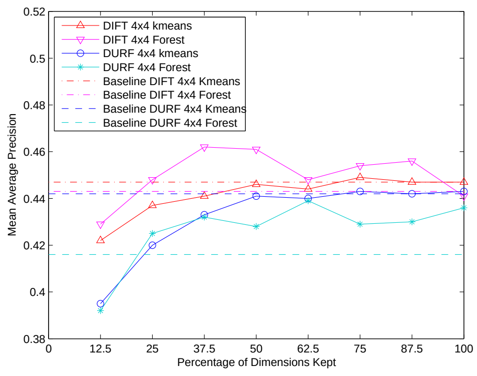
Reduction of the number of dimensions has little influence on accuracy until a certain percentage of dimensions have been reached: for nearest neighbour assignment accuracy stays the same until using half of the total dimensions, for Random Forests accuracy stays the same until using one third of the dimensions. After this point the retrieval performance drops significantly.

At the point where retrieval performance is the same, computation time improves for nearest neighbour assignment with 34% for DIFT 4×4 and 24% for DURF 4×4. At this point computation time for Random Forests with PCA is about 10% slower than Random Forests without PCA step, which is a speed decrease of only 2% with respect to the whole Bag-of-Words pipeline.

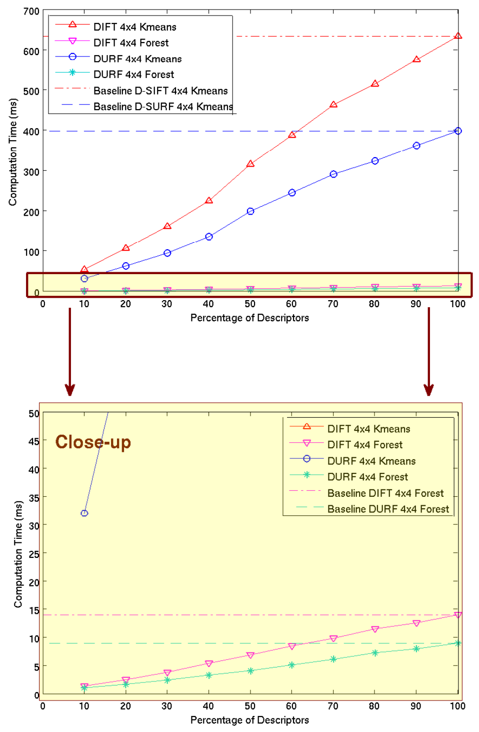
The removal of redundancy in the dimensions by using PCA increases performance for the Random Forest. It implies that



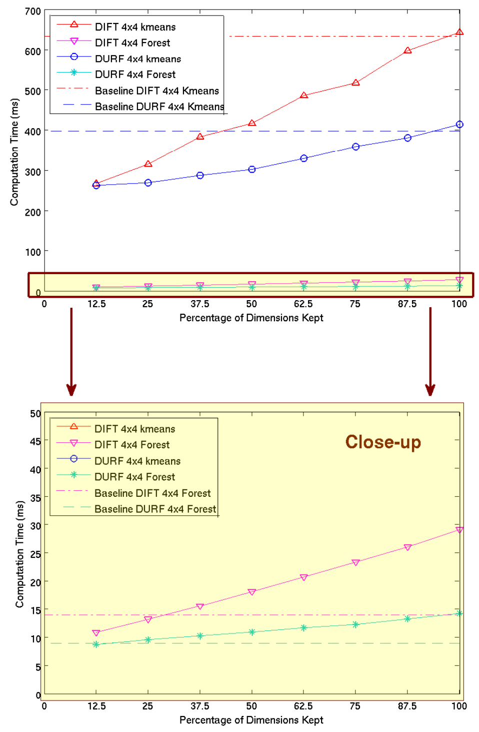
(a) Classification Accuracy Subsampling



(a) Classification Accuracy PCA



(b) Word Assignment Time Subsampling



(b) Word Assignment Time PCA

Fig. 8. Classification Accuracy and Word Assignment Speed when using fewer descriptors of the image. Dashed lines denote the baseline scores.

Fig. 9. Classification Accuracy and Word Assignment Speed when using Principal Component Analysis to reduce the number of dimensions. Dashed lines denote the baseline scores.

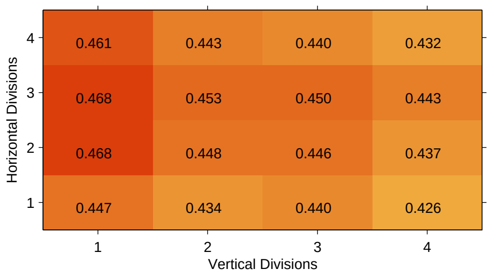
the decorrelation of the dimensions has a positive influence on the decision boundaries of the trees of the Random Forest. This can be understood by the fact that each decision node works on a single dimension. After decorrelation these dimensions contain different rather than overlapping information, making each decision and hence the overall decision better.

Because PCA gives a speed improvement for nearest neighbour assignment and an accuracy improvement for a Random Forest, its use is always beneficial.

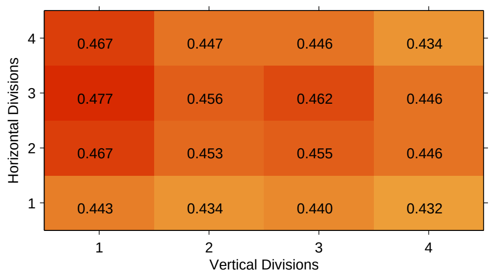
4) Spatial Pyramid: This experiment compares a combination of 1-4 horizontal image regions and 1-4 vertical image regions for the spatial pyramid [8]. The retrieval performance and computational efficiency for DIFT 4×4 for different pyramid divisions is shown in figure 10 and 11 respectively. Results for DURF 4×4 show the same trends.

In terms of retrieval performance, division into 3 horizontal regions is optimal and increases accuracy for all pipelines by 0.02-0.04 MAP. 2 or 4 horizontal regions also give good improvements of around 0.02 MAP. We attribute the increase in accuracy to the crude floor/object/sky distinction it makes.

Our results show that vertical divisions only decrease accuracy in this varied dataset. Even distinguishing the middle part of the image does not lead to extra performance, except marginally for the “horse” and “motorbike” classes, which are

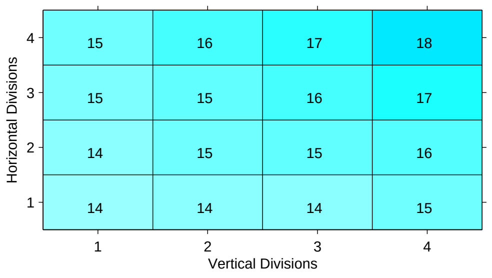


(a) MAP DIFT 4 × 4 k-means

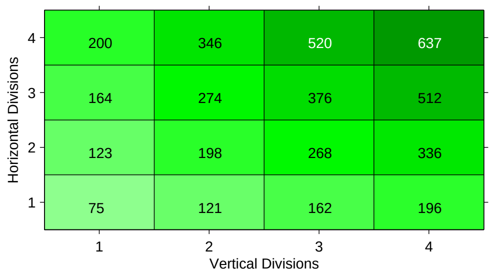


(b) MAP DIFT 4 × 4 Random Forest

Fig. 10. Mean Average Precision scores for several image divisions of the Spatial Pyramid.



(a) DIFT 4 × 4 Random Forest word assignment time



(b) DIFT 4 × 4 Random Forest classification time (χ^2-kernel)

Fig. 11. Speed measurements in milliseconds per image for word assignment and classification for various pipelines and divisions of the Spatial Pyramid.

typical objects of affection.

The word assignment speed is negligibly slower for relevant image divisions, as can be seen in figure 11(a). In contrast, classification time increases considerably as the visual word frequency histograms increase linear in the number of sub-regions. A division into two or three image regions increases computation time with respectively 61% and 116%. Recall that the difference in classification time is only in pre-computing the kernel: afterwards the evaluation of the classification function takes 1 ms for all experiments on this dataset.

To make sure that the increase in retrieval performance is due to the spatial pyramid rather than more statistics, we performed a control experiment in which we increased the visual vocabulary size to 16,384 visual words. For the Random Forest we did this by increasing the depth of the trees to 12. The resulting frequency histogram is as large as a 1×4 or 2×2 subdivision of the image. However, retrieval performance did not increase significantly: For DIFT 4×4 retrieval performance using the large vocabulary is 0.443 MAP for k-means and 0.449 MAP for random forests.

Summarized, the spatial pyramid can be used for good classification improvements at a high computational cost which can not be obtained by simply increasing the visual vocabulary. For this varied dataset only horizontal image divisions should be used.

D. Classification

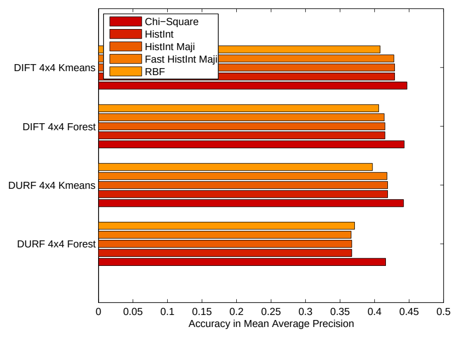
In this experiment we compare the χ^2-kernel, RBF kernel, and the Histogram Intersection kernel. Results are presented in figure 12.

As expected, in terms of retrieval performance the χ^2-kernel is best: its accuracy is about 0.03 MAP higher than the Histogram Intersection kernel and about 0.04 MAP higher than the RBF kernel. The fast, linear approximation of the Histogram Intersection kernel is as accurate as the exact version.

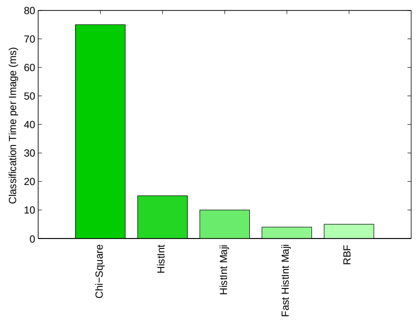
The classification speed of the linear approximation of the Histogram Intersection kernel is best. It is 1800% faster than the χ^2-kernel. The RBF kernel is 1400% faster than the χ^2-kernel. The huge difference between calculating the RBF-kernel and the χ^2-kernel can be attributed to the use of efficient matrix multiplications in calculating the RBF-kernel which is not possible for the χ^2-kernel.

In our experiments the linear approximation of [33] is 300% faster than the precomputed Histogram Intersection kernel, much less than the 5,000-200,000% speed improvement reported in [33]. This is because the pre-computed kernel, which takes up the majority of the classification time, is reused for all 20 classes. Furthermore, the visual word frequency histograms in our pipeline are sparse, which we exploit in our computationally efficient Histogram Intersection implementation.

Looking at figure 12, one notices that using the Histogram Intersection kernel instead of the χ^2-kernel works better when



(a) Classification Accuracy



(b) Classification Time

Fig. 12. Comparison of χ^2 and an Euclidean distance matrix for classification Accuracy and classification speed. Classification speed is measured for a single image for all twenty classes. Classification speed is dependent only on the size of the visual vocabulary which is the same for all methods.

using a k-means visual vocabulary than a Random Forest. Looking at the distribution of visual word frequencies, we observed that the visual word frequencies are more unbalanced for Random Forests than for k-means. Theoretically, the χ^2 distance somewhat normalises this. The Histogram Intersection kernel does not. Therefore we made the visual word counts more balanced by simply taking the square root of these counts. Computational costs are negligible compared to the rest of the pipeline and for DIFT 4×4 and DURF 4×4 this improves the MAP score with 0.01.

To conclude, the fast linear approximation of the Histogram Intersection kernel is the preferred method for computational efficiency as it gives higher accuracy and is faster than the RBF kernel. If the Histogram Intersection kernel is used in combination with the Random Forest it is advised to balance visual word frequency histograms by taking the square root. For optimal classification accuracy the χ^2-kernel is preferred.

E. The Spatial Pyramid Revisited

We now apply the linear approximation of the Histogram Intersection kernel to the horizontal spatial pyramid divisions. Classification results are given in figure 13. Classification time is 6 ms for a division into four subregions, and 4 ms for all other divisions.

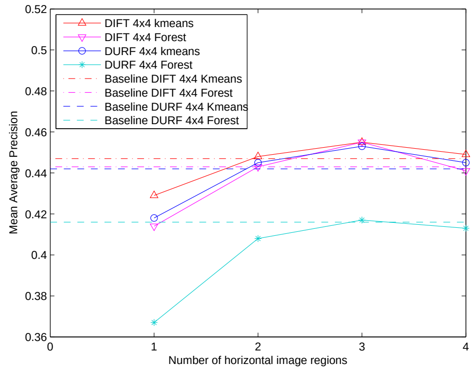


Fig. 13. Retrieval performance for the χ^2-kernel without the spatial pyramid and the fast approximate histogram intersection kernel while using only horizontal divisions of the spatial pyramid.

The retrieval performance of the approximated histogram intersection kernel using three horizontal divisions gives the same or better retrieval scores than the χ^2-kernel baseline without the spatial pyramid. Classification speed at this point is 1800% faster than the baseline.

Theoretically, classification time goes up linear in the size of the word frequency histograms. As we did not observe this trend, we did some extra experiments which showed the caching problems for small word histograms. Hence for small histograms our measurements overestimate the classification times.

Summarized, the approximate Histogram Intersection kernel allows for the inclusion of the Spatial Pyramid at the expense of a small increase of total computation time. It is therefore recommended to include the (horizontal) Spatial Pyramid while using the approximate Histogram Intersection kernel.

F. Supervision in Random Forests

The experiments in this paper provide a very fast Bag-of-Words pipeline which is applicable to large scale datasets in the order of hundreds of thousands of images and beyond. However, large scale datasets tend to come with a large number of classes which are typically not defined from the start. As the Random Forest in our pipeline is created in a supervised manner using the class labels of the images, this begs the question: *If we train a Random Forest on classes other than it is applied, will it still give good results?*

Using the same experimental setup as before, we learn the Random Forest on 5 classes and compare this with the fully supervised Random Forest. Retrieval performance is averaged over 4 trials where we used different classes for learning in each iteration. The results are presented in figure 14.

For DIFT 4×4, using all classes for learning the Random Forest is only 0.007 MAP better than using 5 classes. For DURF 4×4 there is no difference. We furthermore examined the possibility that classification scores were different for the classes on which the tree was learned and on which the tree was not learned. Perhaps surprisingly, this was not the case; the Average Precision scores for the classes on which the tree was not learned was equal to the completely supervised trees.

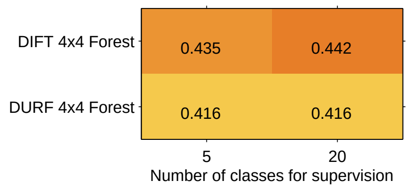


Fig. 14. The influence of supervision of Random Forests on accuracy: A comparison of full supervision with supervision on only 25% of the classes.

We conclude that by using only a few classes for learning one obtains a visual vocabulary which generalizes well to other object classes. This makes Random Forests applicable for datasets where the labels are not defined in advance.

VII. Discussion on Speed

In this paper we have presented various fast alternatives for each of the components of the Bag-of-Words pipeline. In this section we analyse their computational efficiency. We will limit ourselves to methodological improvement. Hardware optimization falls outside the scope of this paper.

A. Descriptor Extraction

For the descriptors used in this paper, descriptor extraction consists of three phases: (1) Apply a gradient filter on a certain scale. (2) Differentiate between positive and negative responses using the absolute value function. (3) Sum responses over a small region.

1) Apply Gradient Filter: For SIFT, we use a fast (approximate) recursive Gaussian derivative filter [43] for the diagonals and an exact derivative filter for the horizontal and vertical directions.

The Haar-wavelets used in SURF are a crude but very fast approximation of the Gaussian derivative filter, using only additions and subtractions of the pixel values itself. By design, the number of computations is easily optimized for a certain scale and resolution.

2) Absolute Value: Taking the absolute value requires a single test per pixel response and is therefore optimal.

3) Sum Responses over Region: We sum responses by using a linear interpolation over subregions by using two matrix multiplications. As we use sparse matrices, the summed responses of a region are calculated using $\mathcal{O}(ab+b)$ additions and the same order of multiplications, where $a \times b$ is the size of a region. This is close to the optimal of $\mathcal{O}(ab)$ additions and multiplications, where our use of existing matrix multiplication libraries gives us an edge over a trivial optimal implementation.

Summing the responses over subregions can be done without any interpolation, in which case only additions are needed over a smaller area, resulting in a small overall speed gain. However, experimenting shows a slight drop in retrieval performance if interpolation is omitted.

B. Word Assignment

We interpret the result of the descriptor extraction phase as follows. Each descriptor which is extracted from an image can be represented as a point in descriptor space. A whole image is represented as a cloud of points in descriptor space, where the clouds of different images can be made up from a varying number of points. In most Bag-of-Words pipelines, the cloud is converted to a k-dimensional vector by dividing the descriptor space into sub-volumes and making a histogram of the number of points per sub-volume. Typically, these sub-volumes are defined by a visual vocabulary created using k-means clustering. These histograms are then compared using a distance measure, which in effect measures the overlap of the cloud. From this perspective, one can understand the work on creating visual vocabularies (*e.g.* [7], [15], [28]) as carving the descriptor space into sensible sub-regions. The work on distance measures or kernels (*e.g.* [18], [27]) can be understood as finding the relative importance of low- and high-density regions.

Alternatively, one could measure the distances between image clouds of descriptors points directly. These distances are then used in a distance based classifier such as a SVM. A known, efficient technique of comparing two clouds directly is the Earth Mover Distance which has an empirical complexity of around $\mathcal{O}(n^3 \log n)$ [45]. This technique is successfully applied to Bag-of-Words by [18], but to make this computationally feasible they first compress each descriptor cloud to 50 points using k-means clustering. However, [18] report that calculating distances using nearest neighbour assignment and the χ^2 kernel is faster and gives comparable accuracy.

In this paper we use the Random Forest as a fast algorithm to project a cloud in descriptor space onto a k-dimensional vector. The Random Forest is a binary decision tree using $\mathcal{O}(n \log k)$ operations for word assignment, where n is the number of descriptors. In contrast, word assignment using a visual vocabulary with nearest neighbour assignment has a considerable larger complexity of $\mathcal{O}(nkd)$, where d is the dimensionality of the descriptor.

An alternative fast word assignment algorithm is a hash function [46] which is of complexity $\mathcal{O}(nd)$. But as in our experiments $\log k < d$, the Random Forest uses less operations and is therefore faster.

C. Classification

The fastest classifier we use is the linear approximation of the Histogram Intersection kernel which has a computational complexity of order $\mathcal{O}(k)$. The χ^2-kernel results in a classification complexity of order $\mathcal{O}(km)$, where m is the number of support vectors.

A fast alternative would be tree-based classifiers which have a complexity linear with respect to the depth of the tree and the number of trees used. Such a classifier would be faster only if the total number of decisions is less than k. It is an interesting research question whether tree-based classifiers are able to perform as well as the Histogram Intersection based SVM, where we observe that a tree-based classifier works well only if there are a few highly informative visual words. In contrast, the SVM works also well if there are large numbers of visual words with low information.

VIII. Conclusions

This paper presented an evaluation of fast Bag-of-Words components to accelerate visual concept classification. We presented results on the Pascal dataset and validated them on the MediaMill Challenge and 15 Natural Scenes databases. Results are similar for all datasets and therefore dataset independent. The presented evaluation leads us to recommend two Bag-of-Words pipelines, one which emphasises accuracy and one which emphasises computational efficiency.

For high accuracy at a minimal computational effort we recommend the Bag-of-Words pipeline shown in figure 15, which consists of DIFT 4×4 descriptors, Random Forest assignment in combination with PCA, a two-level Spatial Pyramid using only horizontal subregions, and the χ^2 based SVM. On the Pascal dataset this pipeline computes classification scores in 297 ms per image with a MAP of 0.501, which is 5% more accurate and 7 times faster than the traditional scheme in figure 2.

If speed is essential, we recommend the Bag-of-Words pipeline shown in figure 16, which consists of DURF 4×4 descriptors, Random Forest assignment in combination with PCA, a two-level Spatial Pyramid using only horizontal subregions, balancing visual word counts using the square root, and the fast approximate Histogram Intersection based SVM. On the Pascal dataset this pipeline computes classification scores in 30 ms per image with a MAP of 0.464, which is 3% less accurate than the traditional scheme but 69 times faster. The discussion of section VII shows that this pipeline is close to optimal in terms of computational efficiency.

Results for the MediaMill Challenge and Natural Scenes database for these two pipelines are presented in table II. Descriptor extraction and word assignment is faster due to smaller image sizes. For the accurate pipeline the classification times are largely dependent on the size of the training set which in the MediaMill challenge becomes a serious bottleneck. After calculating the kernel matrix the classification function itself takes less than 1 ms for the 15 Natural Scenes database. For the MediaMill Challenge this classification function takes 117 ms, which takes so long because of the increased number of support vectors, larger memory access and five times as many classes. In contrast, for the fast pipeline classification times are approximately the same if taken per image per class and scale linearly in the number of classes. The difference in classification accuracy between these two pipelines is comparable for all datasets.

The increased computational efficiency of the Bag-of-Words pipeline in figure 16 opens up new applications for automatic visual concept classification: One application is in the domain of television. The system operates in real-time at a rate of 33 frames per second on a single desktop PC. This enables the tagging of all television of a single channel as it is broadcasted. Another application is in the domain of large image databases. Using 5 computers, this pipeline is able to tag 10,000 images per minute for 20 classes, where each additional computer allows the tagging of 20 extra classes. This throughput is sufficient to automatically tag all pictures that are uploaded to Flickr.

References

[1] C. G. M. Snoek and M. Worring, “Concept-based video retrieval,” *Foundations and Trends in Information Retrieval*, vol. 4, no. 2, pp. 215–322, 2009.

[2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004.

[3] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *IEEE International Conference on Computer VIsion*, 2003.

[4] A. Smeaton, P. Over, and W. Kraaij, “Evaluation campaigns and TRECVID,” in *ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, 2006.

[5] M. Everingham and J. Winn, “The PASCAL visual object classes challenge 2007 (VOC 2007) development kit,” University of Leeds, Tech. Rep., 2007.

[6] C. G. M. Snoek, K. E. A. van de Sande, O. de Rooij, B. Huurnink, J. C. van Gemert, J. R. R. Uijlings, J. He, X. Li, I. Everts, V. Nedović, M. van Liempt, R. van Balen, F. Yan, M. A. Tahir, K. Mikolajczyk, J. Kittler, M. de Rijke, J.-M. Geusebroek, T. Gevers, M. Worring, A. W. M. Smeulders, and D. C. Koelma, “The MediaMill TRECVID 2008 semantic video search engine,” in *Proceedings of the 6th TRECVID Workshop*, Gaithersburg, USA, November 2008.

[7] F. Moosmann, E. Nowak, and F. Jurie, “Randomized clustering forests for image classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 1632–1646, 2008.

[8] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[9] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[11] J. Uijlings, A. Smeulders, and R. Scha, “Real-time bag-of-words, approximately,” in *ACM International Conference on Image and Video Retrieval*, 2009.

[12] M. Marszałek, C. Schmid, H. Harzallah, and J. van de Weijer, “Learning representations for visual object class recognition,” ICCV Pascal VOC 2007 challenge workshop., 2007.

[13] M. Tahir, K. van de Sande, J. Uijlings, F. Yan, X. Li, K. Mikolajczyk, J. Kittler, T. Gevers, and A. Smeulders, “Uva and surrey @ pascal voc 2008,” ECCV Pascal VOC 2008 challenge workshop., 2008.

[14] J. Šochman and J. Matas, “Learning a fast emulator of a binary decision process,” in *Asian Conference on Computer Vision*, 2007.

[15] F. Jurie and B. Triggs, “Creating efficient codebooks for visual recognition,” in *IEEE International Conference on Computer Vision*, 2005.

[16] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[17] K. Mikolajczyk, B. Leibe, and B. Schiele, “Local features for object class recognition,” in *IEEE International Conference on Computer Vision*, 2005.

[18] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, “Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study,” *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[19] M. Grabner, H. Grabner, and H. Bischof, “Fast approximated SIFT,” in *Asian Conference on Computer Vision*, 2006.

[20] E. Tola, V. Lepetit, and P. Fua, “A fast local descriptor for dense matching,” in *IEEE Conference of Computer Vision and Pattern Recognition*, 2008.

[21] J. Shotton, M. Johnson, and R. Cipolla, “Semantic texton forests for image categorization and segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[22] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek, “A comparison of color features for visual concept classification,” in *ACM International Conference on Image and Video Retrieval*, 2008.

[23] J. Willamowski, D. Arregui, G. Csurka, C. Dance, and L. Fan, “Categorizing nine visual classes using local appearance descriptors,” in *ICPR Workshop on Learning for Adaptable Visual Systems*, 2004.

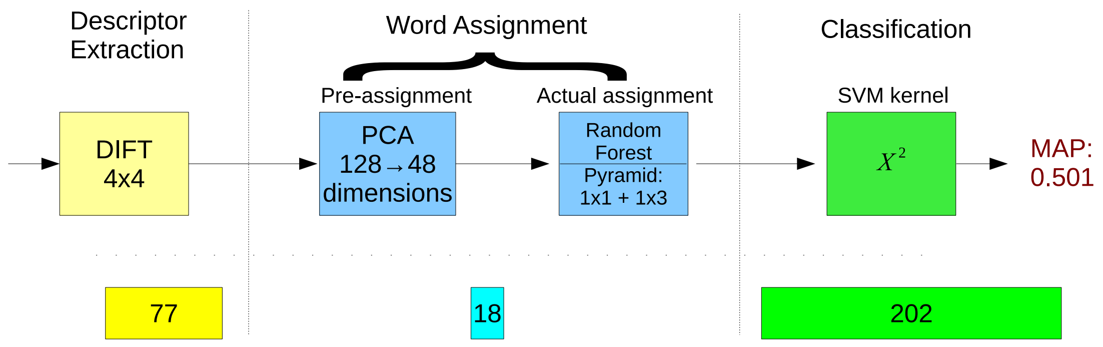


Fig. 15. The recommended Bag-of-Words pipeline when the focus is on accuracy. The accuracy is 0.501 MAP. The classification time is 297 ms, 600% faster and 0.025 MAP more accurate than a traditional Bag-of-Words pipeline (see figure 2).

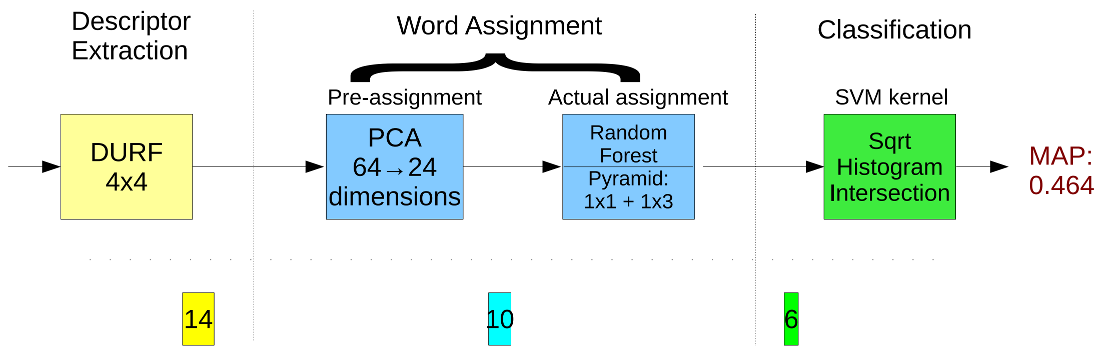


Fig. 16. The recommended Bag-of-Words pipeline when the focus is on speed. Total classification time is 30 ms per image, or 33 frames per second. The accuracy is 0.464 MAP. This is 0.01 MAP less accurate yet 6800% faster than a traditional Bag-of-Words pipeline (see figure 2).

	DIFT	PCA-RF-SP	χ^2	Accuracy
Pascal VOC	77	18	202	0.501
15 Natural Scenes	28	8	104	0.826
MediaMill Challenge	36	9	1638	0.409

	DURF	PCA-RF-SP	Fast HI	Accuracy
Pascal VOC	14	10	6	0.464
15 Natural Scenes	5	7	6	0.778
MediaMill Challenge	6	7	26	0.375

TABLE II
A COMPARISON OF THE ACCURATE AND FAST PIPELINE FOR THE PASCAL VOC DATASET, THE NATURAL SCENE DATABASE, AND THE MEDIAMILL CHALLENGE. NOTICE THAT FOR THE FAST PIPELINE THE CLASSIFICATION TIME CAN BE DIVIDED BY THE NUMBER OF CLASSES. FOR THE ACCURATE PIPELINE THE CLASSIFICATION TIME MOSTLY REFLECTS THE CALCULATION OF THE KERNEL MATRIX.

[24] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[25] J. Gemert, C. Veenman, A. Smeulders, and J. Geusebroek, “Visual word ambiguity,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. in press, 2010.

[26] E. Nowak, F. Jurie, and B. Triggs, “Sampling Strategies for Bag-of-Features Image Classification,” in *European Conference on Computer Vision (ECCV)*, vol. 3954. Springer, 2006, p. 490.

[27] Y. Jiang, C. Ngo, and J. Yang, “Towards optimal bag-of-features for object categorization and semantic video retrieval,” in *ACM International Conference on Image and Video Retrieval*, 2007.

[28] L. Wang, L. Zhou, and C. Shen, “A fast algorithm for creating a compact and discriminative visual codebook,” in *European Conference on Computer Vision*, 2008.

[29] K. Mikolajczyk and J. Matas, “Improving descriptors for fast tree matching by optimal linear projection,” in *IEEE International Conference on Computer Vision*, 2007.

[30] F. Moosmann, B. Triggs, and F. Jurie, “Fast discriminative visual codebooks using randomized clustering forests,” in *Neural Information Processing Systems*, 2006, pp. 985–992.

[31] D. Nistér and H. Stewénius, “Scalable recognition with a vocabulary tree,” in *IEEE Conference on Computer Vision and Pattern Recognition*, New York, 2006.

[32] J. He, S. Chang, and L. Xie, “Fast kernel learning for spatial pyramid matching,” in *IEEE CVPR*, 2008.

[33] S. Maji, A. Berg, and J. Malik, “Classification using intersection kernel support vector machines is efficient,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[34] Nvidia, “CUDA programming guide,” 2010, available at http://www.nvidia.com/CUDA.

[35] T. Sharp, “Implementing decision trees and forests on a gpu,” in *European Conference on Computer Vision*, 2008.

[36] B. Catanzaro, N. Sundaram, and K. Keutzer, “Fast support vector machine training and classification on graphics processors,” in *International Conference on Machine learning*, 2008.

[37] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[38] C. Bishop, *Pattern Recognition and Machine Intelligence*. Springer Science+Business Media, LLC, 2006.

[39] C. G. M. Snoek, M. Worring, J. Gemert, J. Geusebroek, and A. Smeulders, “The challenge problem for automated detection of 101 semantic concepts in multimedia.” in *ACM International Conference on Multimedia*, 2006.

[40] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra, “Video event classification using bag of words and string kernels,” *Image Analysis and Processing*, vol. 5716, pp. 170–178, 2009.

[41] L. Fei-Fei and P. Perona, “A Bayesian hierarchical model for learning natural scene categories,” *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 524–531, 2005.

[42] A. Oliva and A. Torralba, “Modeling the shape of the scene: A

holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[43] J. M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," *IEEE Transactions on Image Processing*, vol. 12, pp. 938–943, 2003.

[44] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001.

[45] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[46] T. Tuytelaars and C. Schmid, "Vector quantizing feature space with a regular lattice," in *IEEE International Conference on Computer Vision*, 2007.