Improving Descriptors for Fast Tree Matching by Optimal Linear Projection

Krystian Mikolajczyk
University of Surrey
Guildford,UK
K.Mikolajczyk@surrey.ac.uk

Jiri Matas
Czech Technical University
Prague, Czech Republic
matas@cmp.felk.cvut.cz

Abstract

In this paper we propose to transform an image descriptor so that nearest neighbor (NN) search for correspondences becomes the optimal matching strategy under the assumption that inter-image deviations of corresponding descriptors have Gaussian distribution. The Euclidean NN in the transformed domain corresponds to the NN according to a truncated Mahalanobis metric in the original descriptor space. We provide theoretical justification for the proposed approach and show experimentally that the transformation allows a significant dimensionality reduction and improves matching performance of a state-of-the art SIFT *descriptor. We observe consistent improvement in precision-recall and speed of fast matching in tree structures at the expense of little overhead for projecting the descriptors into transformed space. In the context of* SIFT *vs. transformed* M-SIFT *comparison, tree search structures are evaluated according to different criteria and query types. All search tree experiments confirm that transformed* M-SIFT *performs better than the original* SIFT.

1. Introduction

In 1999, D. Lowe published a paper [20] describing an object recognition method that integrated three components: (1) a scale-covariant detector of interest regions based on the Difference-of-Gaussian filter, (2) the SIFT descriptor[1], an invariant and stable representation of region appearance by a collection of weighted histograms of gradient orientations and (3) a fast matcher based on kd-tree search for establishing local correspondences. Since then, these components have been used in many state-of-the-art solutions for vision problems that require local image-to-image correspondences, such as wide-baseline matching[2], panorama building [9], image retrieval [21, 27, 29] and object recognition [23].

SIFT represents the state-of-the-art and yet few of its standard parameter choices have a theoretical justification, so it is reasonable to expect that their optimization will lead to a descriptor with performance superior to the state-of-the-art, e.g. with higher repeatability or discriminative power. This idea has been recently explored in [8].

In the paper, we propose to transform the SIFT space so that the nearest neighbor (NN) search for correspondences in step (3) becomes the optimal matching strategy under the assumption that inter-image deviations of SIFT descriptors have Gaussian distribution. This assumption might not be fully satisfied, but it is more realistic than the assumption of isotropic Gaussian noise implicit in standard SIFT matching. The Euclidean NN in the M-SIFT space corresponds to the NN according to a truncated Mahalanobis metric in the original SIFT space. We carry out an extensive evaluation which shows that the transformed descriptor is (i) more discriminative than the original one, (ii) 3-4 times more memory efficient, (iii) no computational overhead, as it does not require spatial Gaussian weighting of image measurements within a patch. These properties are important since memory and efficiency are the limiting factors of large scale recognition methods [27].

The possibility to match SIFT-like descriptors by sublinear tree-based search methods is one of its key properties and has been explored in different systems [18, 21, 23, 27]. We therefore carefully compare SIFT and M-SIFT performance with different search structures, variants of kd-trees and metric (ball) trees. Besides demonstrating that M-SIFT outperforms SIFT, valuable insight into relative merits of different search trees in the context of wide-baseline matching, object recognition and categorization is obtained.

SIFT related work. The idea of representing image parts by histograms of gradient locations and orientations has been used in biologically plausible vision systems [6] and in recognition [2]. Lowe's [20] SIFT is a similar descriptor which performs best in the context of matching and recognition [24]. Several attempts to improve the SIFT have been reported in the literature [3, 10, 15, 17, 24].

Ke and Sukthankar [15] developed the PCA-SIFT descriptor which represents local appearance by principal components of the normalized gradient field. Different performance results were reported for PCA-SIFT on various test data [15, 24], and it is also significantly slower than SIFT to compute. Mikolajczyk and Schmid [24] mod-

[1]Terminological remark. SIFT stands for "Scale-Invariant Feature Transform". Lowe [21] uses the term to refer to the combination of steps (1) and (2). In the literature, the term SIFT often refers just to the descriptor, as e.g. in [24, **?**]. We use the term SIFT in the latter sense.

[2]E.g., most of the top competitors in the ICCV 2005 Computer Vision Contest (http://research.microsoft.com/iccv2005/contest) used SIFT.

ified the SIFT descriptor by changing the gradient location-orientation grid as well as the quantization parameters of the histograms. The descriptor was then projected to lower dimensional space with PCA. This modification slightly improved performance in matching tests. Dalal and Triggs [10] proposed 'histogram of gradients' (HOG). The HOG differs from the SIFT in technicalities like normalization and spatial distribution as well as size of the bins in its many variants. Lazebnik et al. [17] proposed a rotation invariant version called RIFT, which overcomes the problem of dominant gradient orientation estimation required by SIFT, at a cost of lower discriminative power. Bay at al. [3] proposed an efficient implementation of SIFT by applying the integral image to compute image derivatives and quantizing the gradient orientations in a small number of histogram bins. Winder and Brown [8] learn optimal parameter setting on a large training set to maximize the matching performance.

In summary, the modifications of the SIFT are driven by requirements of particular applications and striking different performance trade-offs. The modified versions outperform the original one in some tests, but do worse in others. So far, no method has emerged that would replace SIFT as the descriptor of choice in a wide range of application. The promising strategy seems to be a method for learning descriptors for a particular application rather than designing a general descriptor for any application.

Tree structure related work. Tree structures have been frequently used to handle the problem of fast descriptor matching. A vast number of data structures for fast nearest neighbor (NN) search has been proposed in data mining literature. We focus on two categories of search trees, which have been recently applied in context of image recognition like kd-trees [4, 21, 26] and metric trees [18, 23, 27]. In our experiments, we simultaneously compare different search methods as well as the performance of SIFT and M-SIFT.

Kd-tree [12] belongs to a category of geometric data structures based on hierarchical decomposition of space in multidimensional rectangles. It is frequently reported in the literature that kd-trees are inefficient for dimensions larger than 10. Given that the most powerful descriptors used in vision have many more dimensions, one would not consider kd-tree as a possible solution. However, an approach to overcome the dimensionality problem is to search for approximate NN [1, 19]. Vision applications have found this solution to be sufficient [21, 26] as other parts of the recognition systems are highly inaccurate too. A modified kd-tree with priority queue and approximate NN search in 128 dimensional space was successfully used for fast matching in [21]. Superior performance of this structure over ball-tree [25] is reported in [16]. This indicates that in some specific problems kd-tree can provide satisfying solutions. However, it is still unclear whether kd-tree outperforms other methods in matching image descriptors. One of the extensions to handle the dimensionality problem in kd-tree was proposed in [1] based on spatial decomposition to guarantee exponential cardinality of points and geometric reduction of node sizes as descending the tree.

Metric based indexing methods seem to be more suitable for general NN search problem in high dimensional spaces. Generalized hyperplane partitioning for building binary metric trees, termed gh-tree, was introduced in [30] and extended in [7]. Simplified versions of such trees were used in [18, 23, 27] for matching image descriptors. More detailed review of metric trees and other index structures can be found in [5, 14].

In our experiments we use several speed up techniques to extend the trees proposed in [18, 27] and provide extensive evaluation of SIFT and M-SIFT features.

Overview. The rest of the paper is organized as follows. In section 2 we define the context of this work and basic assumptions. Section 3 describes our method for improving image descriptors. Section 4 revises data structures evaluated in this paper. Finally, section 5 presents and discusses the experimental results.

2. Correspondence by SIFT Matching

Our objective is to transform descriptors to improve their matching performance. More precisely, we aim at reduction of the percentage of false matches among the tentative correspondence formed by descriptor matching while preserving the desirable property of speed of computation and compact representation. To this end, we adopt the following model of the matching process.

Two sets of features $\mathcal{F}_I$ and $\mathcal{F}_D$[3] are given. In different computer vision problems, the two sets appear under different names, but play essentially identical roles. In object recognition, $\mathcal{F}_I$ is the set of descriptors detected in the test image and $\mathcal{F}_D$ the set of descriptors in the database, representing objects acquired in the training stage. In wide-baseline matching, $\mathcal{F}_I$ and $\mathcal{F}_D$ represent features detected in the left and right images respectively. Panorama stitching may be formulated as either repeated wide-baseline matching or a recognition problem.

The problem of finding correspondences (matching regions) is defined as a search for a function $B : \mathcal{F}_I \rightarrow \mathcal{F}_D \cup \mathbf{x}'_0$ that assigns to every $\mathbf{x_i} \in \mathcal{F}_I$ either a $\mathbf{x}'_\mathbf{j}$ from the database $\mathcal{F}_D$ or $\mathbf{x}'_0$ representing no match. Three remarks about this formulation are in order. First, symmetric wide-baseline matching can be easily formulated by either performing the matching twice, exchanging left and right images, or by defining both $\mathcal{F}_I$ and $\mathcal{F}_D$ as unions of descriptors from both images and constraining B to avoid matching two points from the same image. Both options

[3]Notation. Different fonts are used to distinguish sets $\mathcal{S}$, vectors $\mathbf{x}$, matrices $\mathtt{C}$ and functions B, g, λ. Symbol $\|.\|_2$ denotes the Euclidean norm, $P(.|S)$ conditional probability.

are conceptually simple, but make notation more complicated; for brevity, we therefore use the asymmetric formulation appropriate for the object recognition problem. Second, the formulation ignores the fact that the mapping B should be, with the exception of the 'no match' element $\mathbf{x}'_0$, one-to-one (a bijection), respecting the constraint that no $\mathbf{x}'_\mathbf{j} \in \mathcal{F}_D \backslash \mathbf{x}'_0$ matches more than one feature from $\mathcal{F}_I$ and vice verse. Enforcing the one-to-one constraint makes the decision whether $\mathbf{x}_\mathbf{i}$ is assigned to $\mathbf{x}'_\mathbf{j}$ dependent on every single observation in $\mathcal{F}_I \cup \mathcal{F}_D$. Matching algorithms that respect this uniqueness have computational complexity much higher than the simple kd-tree nearest neighbor algorithm and are therefore not considered. We assume, in line with common practice, that uniqueness among the tentative correspondences is enforced ex post. Finally, we note that by positing $\mathcal{F}_D \cup \mathbf{x}'_0$ as the domain of B, we have implicitly defined that a region can be matched to either zero or one counterpart. However, the probabilistic model presented below can be easily extended to situation where B maps to $2^{\mathcal{F}_D}$, i.e. where the matching procedure associates with each feature of the input image a (possibly empty) subset of $\mathcal{F}_D$ (see sec. 5).

The standard procedure of SIFT matching (step (3) of Lowe's method) [20] assigns to $\mathbf{x}_\mathbf{i}$ its Euclidean nearest neighbor:[4]

$$B_L(\mathbf{x}_\mathbf{i}) = \arg \min_{\mathbf{x}'_\mathbf{j} \in \mathcal{F}_D} \|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2. \tag{1}$$

For the chosen formulation of the matching process, the optimal procedure from the point of view of generating a maximum number of true correspondence among the established correspondences is based on the likelihood ratio

$$\lambda(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}) = \frac{P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|S)}{P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|\bar{S})}, \tag{2}$$

where $P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|S)$ is the probability that the two observations $\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}$ correspond to two views of the same surface patch S, $P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|\bar{S})$ is the probability that their pre-images are different. In other words, $P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|S)$ models inter-image variations of appearance of patch S, $P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|\bar{S})$ expresses natural image statistics. Rule (2) justifies the B_L assignment of Eq. (1) under the following condition

$$P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|S) = P(\|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2|S) = g(\|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2),$$

where g(.) is a monotonic function. The obvious, but not unique, case is $P(\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}|S) \sim N(0, \text{diag}(\sigma))$, i.e. $P(.|S)$ has a zero-mean isotropic Gaussian distribution and $g(x) = e^{-x}$. The B_L model thus implicitly states: (i) the probability that two descriptors correspond to the same surface patch depends only on their difference and (ii) the inter-image difference has isotropic Gaussian distribution; the value of σ is irrelevant.

[4]This is a simplified description which also applies to any other descriptor. The two most important technicalities omitted are: (i) no match is assigned if the second nearest point from $\mathbf{x}_\mathbf{i}$ is not far enough, i.e. if $\min_{\mathbf{x}'_\mathbf{j} \in \mathcal{F}_D} \|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2 / \min_{\mathbf{x}'_\mathbf{j} \in \mathcal{F}_D \backslash B_L(\mathbf{x}_\mathbf{i})} \|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2 > \epsilon$, default $\epsilon = 0.8$ and (ii) the min operation is carried out only approximately.

3. MSIFT: Mahalanobis SIFT

We discuss the descriptor transformation using SIFT but the procedure is applicable to any other descriptor. We model the probability that two patches are in correspondence as

$$P(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{j}|S) = P(\|\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2|S) = N(0, \mathtt{C}_\mathtt{S}), \tag{3}$$

where $\mathtt{C}_\mathtt{S}$ is a covariance matrix with rank D. The subscript S indicates that $\mathtt{C}_\mathtt{S}$ models variations of observations of the same Surface. The mean of the distribution is zero, since it is a difference of two identically distributed random variables. Estimating full-rank $\mathtt{C}$ is feasible on a large training set of corresponding SIFT pairs. Nevertheless, there are several reasons to choose the dimensionality D of the covariance matrix $\mathtt{C}$ as low as possible. Firstly, D defines the number of dot products that transform SIFT to M-SIFT. Secondly, $D/128$ is the fraction of memory required by M-SIFT compared to 128-dimensional SIFT. Moreover, M-SIFT matching performance peaks for values of D that are only a fraction of 128.

To be able to benefit from the efficiency of the fast NN methods and yet to take into account the approximation of the inter-image variability of region appearance modeled by Eq. 3 we apply a whitening linear transform of the SIFT space. The whitening transform is not defined uniquely and we choose the linear transformation so that the new space facilitates dimensionality reduction. Details about the linear transformation are given next.

Computing M-SIFT by simultaneous diagonalization. We collected a training sets of difference vectors $\mathcal{T}_S$ and $\mathcal{T}_{\bar{S}}$. Difference vectors are in $\mathcal{T}_S$ if they have the same pre-image. Estimates of the covariance matrices $\mathtt{C}_\mathtt{S}$ and $\mathtt{C}_{\bar{\mathtt{S}}}$ are computed using the training sets. The linear transformation $\mathtt{T}$ that is applied to the SIFT space is the inverse of the square root of the 'same surface' covariance matrix $\mathtt{C}_\mathtt{S}$: $\mathtt{T} = \mathtt{C}_\mathtt{S}{}^{-\frac{1}{2}}$. We will denote vectors in the transformed space as $\mathbf{y}_\mathbf{i} = T\mathbf{x}_\mathbf{i}$. In the $\mathbf{y}$-space, Euclidean distance is a monotonic function of the probability that the two description vectors involved originate from the same surface patch:

$$\mathtt{C}^\mathbf{y}_\mathtt{S} = \sum_{(\mathbf{x}_\mathbf{i}, \mathbf{x}'_\mathbf{i}) \in \mathcal{T}_S} \mathtt{T}(\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{i})(\mathbf{x}_\mathbf{i} - \mathbf{x}'_\mathbf{i})^\top \mathtt{T}^\top = \mathtt{T}\mathtt{C}_\mathtt{S}\mathtt{T}^\top = \mathtt{1}.$$

The diagonality of $\mathtt{C}^\mathbf{y}_\mathtt{S}$ is preserved if (i) the $\mathbf{y}$-space is rotated, i.e. transformed by any orthonormal matrix $\mathtt{R}, \mathtt{R}\mathtt{R}^\mathtt{T} = 1$ and (ii) if some dimension are dropped. We use these two properties to reduce the dimensionality of the transformed descriptor space. A second transformation is applied, aligning the axis with the eigenvectors of the covariance matrix $\mathtt{C}^\mathbf{y}_{\bar{\mathtt{S}}}$ modeling the variation of two randomly chosen patches. The transformation is a rotation, since a symmetric matrix like $\mathtt{C}^\mathbf{y}_{\bar{\mathtt{S}}}$ has orthonormal eigenvectors. Finally, we drop $128 - D$ dimensions with the smallest variance; D is established experimentally. The final projection into the M-SIFT space is a matrix multiplication by $D \times 128$ matrix

$$\mathtt{P} = \mathtt{D}\mathtt{R}\mathtt{T}, \tag{4}$$

where $\mathtt{R}$ is the matrix of eigenvectors of $\mathtt{C}^{\mathbf{y}}_{\bar{\mathtt{S}}}$ sorted by eigenvalue magnitude and $\mathtt{D}$ is a selector or the first D-rows.

The procedure can be summarized as whitening of $\mathtt{C}_{\mathtt{S}}$ within a simultaneous diagonalization [13] of $\mathtt{C}_{\mathtt{S}}$ and $\mathtt{C}_{\bar{\mathtt{S}}}$, followed by a PCA. (The whitening is applied so that the data match the Euclidean nearest neighbor rule. The dimensionality reduction part removes directions in the SIFT space not contributing to the match.

4. Efficient Matching

Our approach modifies the descriptor to achieve better matching performance in different data structures. Before we present the experiments, we briefly revise the data structures recently used for matching interest point descriptors in the context of visual recognition. We then explain search strategies and pruning techniques, which we apply to improve the matching used in [4, 18, 21, 27].

4.1. Tree structures

Kd-tree. The recognition algorithm used in [4, 21] is based on kd-tree structure. The tree is created by iterative partitioning. At each iteration, the data set is split at the median of the dimension of largest variance. This creates a balanced binary tree with depth $\log_2 N$ (cf. Fig. 1). However, the aspect ratio of rectangular cells of the tree is not in general bounded, e.g. the cells may be very long in one dimension and short in others. Consequently, during search a query sphere can intersect many such elongated cells. A modification of kd-tree called balanced box-decomposition tree was proposed in [1]. Its spatial decomposition guarantees exponential decrease of the cardinality of points and geometric reduction of node sizes as descending the tree. They define a cell by box or the set theoretic difference of two boxes, outer box and an optional inner box. To obtain even partitions, two splits are applied alternately: a regular hyperplane split and a shrinking split which uses a box rather than a hyperplane. Thus an evenly partitioned balanced tree with bounded aspect ratio of cells is constructed. Fig. 1 illustrates the hyperplane splits in blue and box partitions in red.

Metric tree. A ball tree (or metric tree) is a hierarchical structure for representing a set of points where the balls are regions bounded by a hypersphere in a multidimensional metric space [30]. Each node (ball) of the tree contains several children balls and is represented by the center and radius. The center node is a mean vector of its children leaf nodes, and the radius is determined by the point farthest from the center. Unlike in kd-trees, cells in ball-trees can intersect and do not have to partition the entire space (cf.Fig. 1). There are many methods to build metric trees [28], the approaches can be classified as top-down partitioning, bottom-up agglomerative and middle-out methods. A top-down approach was used in [27] (cf. Fig. 1). K-means clustering with predefined branching factor was applied recursively starting from the top node to partition the data points into children nodes. Each node was represented by a centroid. The main advantage is the efficiency of the partitioning method but the structure suffers from problems intrinsic to k-means, e.g. sensitivity to outliers, imposed K, centroids far from real clusters resulting in large cells. In the bottom-up approach, agglomerative average-link clustering was used to create the tree [18] (cf. Fig 1). The algorithm starts with the leaf nodes centered at each data point. At each iteration two nearest nodes are merged creating a parent node. The method continues until top nodes are merged. A post processing method merges some parents with children to reduce the number of nodes and to form a more compact structure. Compactness, that is the small volume of hyperballs maximizes the number of prunings that may occur during NN search thus makes the search faster [28]. However, the complexity of bottom-up methods makes it impractical for large data sets. Middle-out technique [25] is a trade-off between efficiency and compactness of the tree.

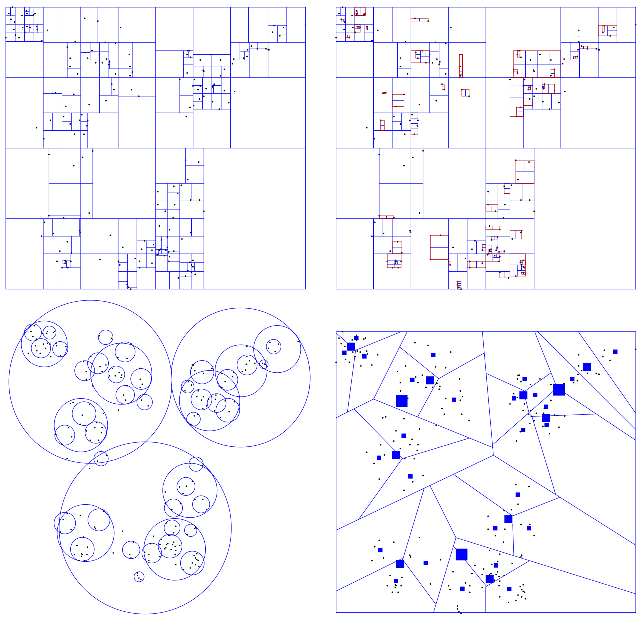
Figure 1. (Top-left) Kd-tree. (Top-right) Bbd-tree. (Bottom-left) Bottom-up ball-tree. (Bottom-right) Top-down kmeans.

4.2. Search strategies

In this section we define different query types evaluated in this paper, which are typically used in matching applications.

Range search. The objective is to find all data points within a given distance from the query. Ball-trees based on Euclidean distance are suitable for such search. The search starts with the top nodes and verifies the query and node intersections. Euclidean distance is computed to every node center and the intersections are inferred given the query and the node radii. All nodes which intersect with the query radius have to be traversed.

kNN search. This method is concerned with finding k nearest neighbors given a query point. If $k = 1$, the search returns a single NN. The tree is first traversed by entering cells which are closest to the query point. The branches are explored until k leaf nodes are found. The distance to the k-th neighbor is then used to bound the search and only nodes which are closer to the query have to be examined.

ANN, approximate kNN search. In high dimensional spaces, a query sphere can intersect with a large numbers of nodes and the search for NN becomes inefficient since all intersecting nodes have to be visited. The number of visited cells can be significantly reduced if the NN search is only approximate. The procedure is similar to exact kNN except that the search is continued while a given criterion is satisfied. The criterion can e.g. constrain the maximum number of leaf nodes to visit. Another possibility is to terminate the search if the distance from the closest cell to the query exceeds $\delta = d(p, q)/(1 + \epsilon)$, where p is the NN found so far, q – query, ϵ is a positive termination parameter. In other words, it guarantees that no subsequent point to be found can be closer to q than δ. In ball-tree structure, the node radius can also be defined by a quantile of ordered leaf distances from the node center. Thus points, which are far from the node center are not considered. This reduces the number of intersections and allows to prune many nodes early on in the search process. These points can be still close to other node centers since the nodes intersect. In all these techniques a certain percentage of returned points are not the exact nearest neighbors. Many parts of visual recognition systems are highly inaccurate therefore approximate search often provides sufficient results.

4.3. Pruning techniques

Tree search methods differ mainly in how the hierarchy is traversed and how the branches are pruned to reduce the number of nodes examined during search. We briefly discuss the methods used in this paper. Further details can be found in [1, 14, 28].

Branch and bound. This method can be used in kNN search and it makes use of the distance between the query and the leaf points found so far to reduce the query radius during the search. This allows to reject all the cells, to which the distance from the query is larger than the distance to kth neighbor found so far.

Triangle inequality. In the case of a metric obeying this inequality, given distance $d(a, b)$ between points a and b, and $d(b, c)$ between points b and c we can compute lower and upper bounds on distance $d(a, c)$, $|d(a, b) - d(b, c)| \leq d(a, c) \leq d(a, b) + d(b, c)$. A node can be rejected if the bounds fall outside of query radius. These pruning criteria can be used in both, the tree construction and the search.

Priority queues. Priority queues are successful in limiting the number of traversed nodes in approximate NN search. Given a termination criterion, we increase the probability of encountering nearest points earlier by examining nodes in order of increasing distance. One possibility is to maintain a priority queue of all encountered nodes and distances to them. A new node from the top of the queue is examined as soon as the tree is traversed down to the leaf node. Another method is to verify the top node from the queue every time the distance to a node is computed. The node, to which the distance is smaller, is traversed.

5. Results

In this section, we present experiments testing the matching performance of the SIFT transformed with the proposed method, using tree data structures. We first describe the evaluation framework and then investigate the performance of the descriptors according to different criteria.

5.1. Experimental framework

We adopt the evaluation criteria and test data proposed in [24], where the best repeatability and region accuracy is achieved by the MSER detector [22]; we therefore use this detector in all experiments. Three variants of the SIFT descriptor are evaluated. The original SIFT serves as a reference, with PCA projections if 40 or 20 dimensions are only used. Next, a SIFT without spatial Gaussian weighting of image gradients within a region, termed U-SIFT, is tested. We also apply the Mahalanobis like normalization to the original SIFT and denote it MO-SIFT. Finally, we include the implementation of M-SIFT which is transformed by $\mathtt{P}$ of Eq. 4, but is not weighted by the 2D spatial Gaussian.

The covariance matrices $\mathtt{C}_\mathtt{S}$ and $\mathtt{C}_{\bar{\mathtt{S}}}$ needed for computation of $\mathtt{P}$ are estimated on an independent set of 30 image pairs [5]. Dimensionality D of M-SIFT and MO-SIFT (the rank of $\mathtt{P}$) is varied and results are presented for different number of dimensions.

We also test the performance of the descriptors in four different data structures for NN search: two kd-tree variants called BBF [4, 21] and BBD [1] and two types of ball-trees; bottom-up BU [18] and top-down TD [27] with speedup improvements.

Test data. Results for the variants of the SIFT descriptor are presented for two sequences with viewpoint and scale change. Results on other sequences from the publicly available set[6] are consistent with those presented here. Each sequence consists of 6 images with gradually increasing transformation and ground truth homographies.

To evaluate matching in data structures, 10^6 features were collected from [11] training images. Furthermore, two

[5]http://lear.inrialpes.fr/people/mikolajczyk/Database
[6] http://www.robots.ox.ac.uk/~vgg/research/affine

sets of 10^3 query points were prepared. ”Near queries”, termed NQ, simulate matching and retrieval scenario. Features from images showing the same scenes but viewed from a different angle and scale were used. ”Far queries”, FQ were collected from images with similar scenes to those in the database. This set simulates matching for category recognition and contains many points which come from a different distribution. Note, that the difference between NQ and FQ is the average distance to their nearest neighbors in the database.

Finally, for testing the recognition performance we use UIUC multi-scale car images and TU-Darmstadt multi-scale motorbikes [11] .

Performance measure. Following the evaluation framework proposed in [24], we match each of the images from a sequence to the reference image. Two features are considered matched if their similarity distance is below a threshold. The match is correct if the spatial overlap of regions projected with the homography is more than 40%. We vary the similarity threshold and obtain a precision-recall curve for each image pair as defined in [24]. In order to present the results for one image sequence in a compact form, we consider the area below a precision-recall curve. One curve represents results for entire image sequence.

Matching efficiency for different data structures is measured with respect to the exhaustive search and the tree matches are compared with those returned by the exhaustive search. If a different point was returned by the tree search, it was counted as an approximate NN. The percentage of ANN was controlled with ϵ based criterion or by limiting the number of examined nodes. The cost of descriptor projections is included in the comparative results. In this evaluation we use the truncated Euclidean distance which accelerated the exhaustive search by a factor of 1.4.

5.2. Results

Matching performance. The results shown in Fig. 2(top) are computed for image pairs with increasing transformation. Performance of the original SIFT is already high on this test data, therefore even small improvements are significant. The figures demonstrates that MO-SIFT performs better than SIFT, which shows that the Mahalanobis like normalization still improves matching. Further improvement given by M-SIFT indicates that there exist a better weighting function than a Gaussian, which can be learnt from the training data. However, the lower results for unweighted descriptor U-SIFT show that the Gaussian window does have a positive impact on the SIFT performance. These observations are consistent for different scenes and image transformations.

Dimensionality. The main advantage of the proposed method is the dimensionality reduction while maintaining the high performance level of the original SIFT.

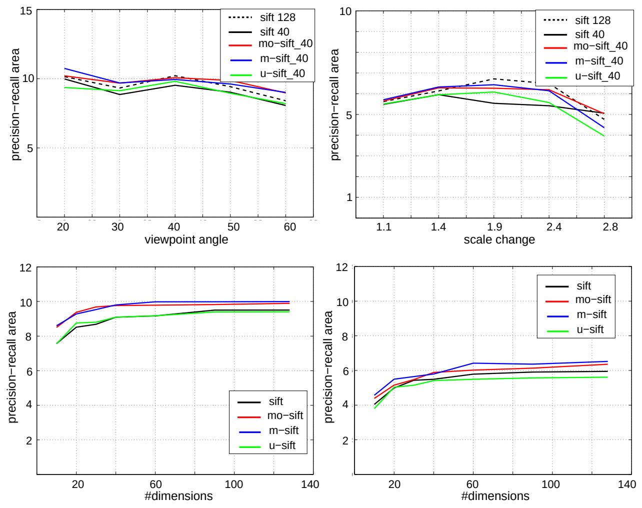

Figure 2. Precision-recall area for sequences. (Left) Viewpoint (Grafitti). (Right) Scale change (Boat). (Top) Transformation. (Bottom) Dimensionality.

Fig. 2(bottom) shows the precision-recall area for different number of dimensions. The results were obtained for 40^o viewpoint angle, and 2.5 scale change. We observe that the performance increases up to 40 dimensions. Using more dimensions does not bring significant improvements. The top score for all sequences is obtained by M-SIFT; MO-SIFT comes second. Unweighted SIFT obtained lower score which again validates the use of a weighting window over the interest region. Thus, the proposed M-SIFT projection results in better performance and lower memory requirements for negligible projection cost.

Search queries. The tree structures were constructed with algorithms proposed in the corresponding publications [1, 4, 18, 27] and the search was done with the improvements discussed in section 4.3. Table 1 shows speedup factors and the construction time compared to exhaustive search. The numbers show that kd-trees have a great advantage over bottom-up metric trees in the efficiency of the tree construction. In the NN search experiments we have obtained significantly different results for near NQ and far FQ queries. The methods were set to return 10% of approximate NN. Search on NQ in kd-tree is extremely fast and accurate. 1NN was found 36000 times faster than with the sequential search in 10^6 128-dimensional features. The NN feature is quite often found in first few investigated leaf nodes and the search terminates very early. The gain in search speed is however much lower for FQ, e.g. 2 for bbf-tree. This discrepancy is smaller for metric trees e.g. 27 for NQ and 4.1 for FQ. In the remaining experiments we provide results for negative far queries, which give a lower bound on matching speed.

Fast search. In this experiment we compare the performance of the proposed descriptor in different data structures. We search for 10NN in 40-dimensional M-SIFT space with FQ. Fig. 3(top-left) shows speed improvements com-

	bbf-tree		bbd-tree		bu-tree		td-tree	
#NN	1	10	1	10	1	10	1	10
NQ 128	33000	83	1500	7	32	27	29	22
NQ 40	27000	8	35	2	12	11	10	9
FQ 40	2.2	2	0.3	0.1	4.4	4.1	3.7	3.3
Train.	22s		185s		47h		625s	

Table 1. Comparison of speedup factors as well as tree construction time for near and far queries in 128 and 40 dimensional space.

pared to exhaustive search for different percentage of ANN. In contrast to the results on NQ, the fastest search for FQ was obtained with the metric trees. BBF kd-tree is on average twice slower that the metric tree built with bottom-up technique BU. BBD kd-tree shows the lowest performance. Range search is noticeably slower than 10NN since no query radius reduction is done during search. Fig. 3(top-right) compares 1NN search with 10NN in 40 and 128-dimensional space. The speedup gain of metric trees is larger for 128 than 40 dimensions, as expected. Interestingly, to obtain less than 20% of ANN in 128 dimensional kd-tree the search is slower than the sequential one which makes the use of kd-tree questionable. In contrast, kd-tree of 40 dimensional points provides an improvement already for 2% of ANN. The benefit of using M-SIFT instead of PCA projected SIFT in both, kd-tree and metric tree, is demonstrated in Fig. 3(bottom-left). The presented experiments show that the proposed descriptor makes possible to use kd-trees in the categorization scenario and gives better results than standard PCA approach.

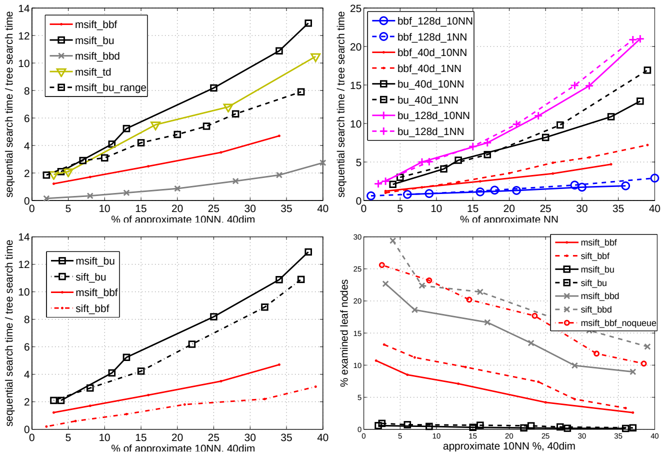

Figure 3. Efficiency of the similarity search. (Top-left) Tree comparison. (Top-right) Dimensionality results. (Bottom-left) Descriptor comparison. (Bottom-right) Priority queues.

Pruning. Priority queues reduce the number of traversed nodes by examining them in order of distance to the query but the overhead for maintaining the queues slows down the search. We observed that, although the number of examined nodes is low compared to *noqueue* search (cf. Fig. 3(bottom-right)) the actual speed decreases by a factor of 2. This indicates that the cost of computing distance is lower than updating and verifying the queue. The previous experiments were therefore done without priority queues. Priority queues can be useful in large databases where the access to cells stored on different hard drives is expensive. However, this may vary for different implementations since inconsistent reports on performance of priority queues can be found in the literature [1, 4, 5, 14, 21]. Fig. 3(bottom-right) shows that the number of examined nodes is consistently smaller for M-SIFT than SIFT in 40-dimensional space, which indicates that similar data points in M-SIFT space are lying closer to each other compared to SIFT. We also observed that limiting the number of examined cells improves the efficiency but the fraction of ANN significantly vary for different queries. Distance based stopping criterion allows to control the level of NN approximation at little expense of speed. Pruning based on triangle inequality improved the search speed by a factor of 1.4. Traversing metric tree to the first leaf node only, as done in [27], results in 99% of ANN for FQ and 45% of ANN for NQ. Although this search technique is extremely fast, the matching quality is rather low. Using multiple trees with randomized partitions may provide improvement as reported in [29].

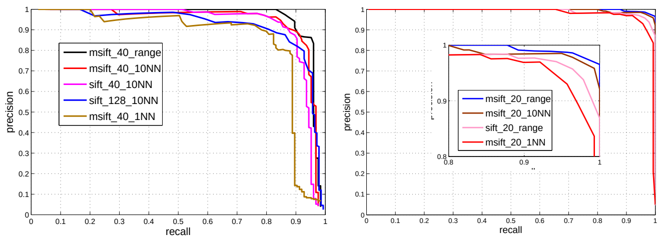

Figure 4. Object recognition results. (Left) TUD-Motorbikes. (Right) UIUC-Cars.

Recognition. In this experiment we compare recognition results obtained with range, 1NN, and 10NN search. We use recognition system from [23], which is based on matching features to a visual vocabulary. The matched visual words indicate positions of objects in a voting space. The search methods were set to return less than 10% of ANNs. Fig. 4(left) and (right) show precision-recall results on multi-scale motorbike test set and multi-scale car test set, both accessible from [11]. Note that although our primary goal here is to compare different matching and search strategies, the recognition score outperforms state-of-the-art results in [11, 23]. This is due to the large number of features sampled from the training data which are efficiently dealt with by trees. The processing of a test image takes less than one second and most of the computation is in the feature extraction. Range search in bottom-up tree of M-SIFT features gives the highest score. Good results are also obtained with 10NN search in contrast to 1NN. Many object instances are correctly detected when 1NN is correct however any matching errors made at these stage are hard to recover later in the recognition process. 10NN search strategy with kd-trees seems to be a good tradeoff to avoid complexity problems of metric trees.

Conclusions and discussion

In this paper we presented a method for improving image descriptors by learning optimal projections. Experimental results show that the approach leads to a significant dimensionality reduction as well as to an improvement of the matching performance. We observe consistent improvement in matching quality and speed of fast tree search.

In addition, we perform an evaluation of tree structures using SIFT and M-SIFT according to different criteria and on different queries. The results show extremely high performance of kd-trees in high dimensional spaces on near queries and very low on far ones. More consistent results are obtained with metric trees although the construction complexity is high. Kd-tree with priority queue and limited number of examined cells seems to be good choice for matching or retrieval of transformed images where corresponding descriptors are relatively close in the feature space. In category recognition problems, range search gives significantly better results than 1NN search. 10NN and kd-tree is a possible trade-off if the complexity of tree construction is an issue. Finally, in all search tree experiments we observed that M-SIFT performs better than SIFT.

Acknowledgements

This research was supported by EU VIDI-Video-045547 project and by Czech Science Foundation project 102/07/1317.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. 2, 4, 5, 6, 7

[2] A. Ashbrook, N. Thacker, P. Rockett, C. Brown. Robust recognition of scaled shapes using pairwise geometric histograms. In *BMVC*, 1995. 1

[3] H. Bay, T. Tuytelaars, L. van Gool. SURF: Speeded up robust features. In *ECCV*, 2006. 1, 2

[4] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR*, 1997. 2, 4, 5, 6, 7

[5] C. Bohm, S. Berchtold, D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001. 2, 7

[6] V. Brecher, R. Bonner, and C. Read. A Model of Human Preattentive Visual Detection of Edge Orientation Anomalies. In *SPIE CVIP*, 1991. 1

[7] S. Brin. Near neighbor search in large metric spaces. In *VLDB*, 1995. 2

[8] S. Winder and M. Brown. Learning Local Image Descriptors. In *CVPR*, 2007. 1, 2

[9] M. Brown and D. Lowe. Recognishing panoramas. In *ICCV*, 2003. 1

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2

[11] M. Everingham and et. al. The 2005 PASCAL Visual Object Classes Challenge. In *PASCAL Challenges Workshop, LNAI*, 2005. 5, 6, 7

[12] J. H. Friedman, J. L. Bentley, R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977. 2

[13] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990. 4

[14] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, 2003. 2, 5, 7

[15] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *CVPR*, 2004. 1

[16] D. Lang, M. Klaas, N. de Freitas. Empirical testing of fast kernel density estimation algorithms. *UBC Technical repor*, 2005. 2

[17] S. Lazebnik, C. Schmid, J. Ponce. A sparse texture representation using local affine regions. In *CVPR*, 2003. 1, 2

[18] B. Leibe, K. Mikolajczyk, B. Schiele. Efficient clustering and matching for object class recognition. In *BMVC*, 2006. 1, 2, 4, 5, 6

[19] T. Liu, A. Moore, A. Gray, K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *NIPS*, 2004. 2

[20] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 1, 3

[21] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 4, 5, 7

[22] J. Matas, O. Chum, U. M., T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002. 5

[23] K. Mikolajczyk, B. Leibe, B. Schiele. Multiple object class detection with a generative model. In *CVPR*, 2006. 1, 2, 7

[24] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE T. PAMI*, 27(10):1615–1630, 2005. 1, 5, 6

[25] A. W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *UAI*, 2000. 2, 4

[26] G. Mori, S. Belongie, H. Malik. Shape contexts enable efficient retrieval of similar shapes. In *CVPR 1*, 2001. 2

[27] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006. 1, 2, 4, 5, 6, 7

[28] S. Omohundro. Five balltree construction algorithms. *Technical Report TR-89-063*, 1989. 4, 5

[29] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *In CVPR*, 2007. 1, 7

[30] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inf. Process. Lett.*, 40(4):175–179, 1991. 2, 4