Learning Local Image Descriptors

Simon A. J. Winder Matthew Brown
Microsoft Research
1 Microsoft Way, Redmond, WA 98052, USA
{swinder, brown}@microsoft.com

Abstract

In this paper we study interest point descriptors for image matching and 3D reconstruction. We examine the building blocks of descriptor algorithms and evaluate numerous combinations of components. Various published descriptors such as SIFT, GLOH, and Spin Images can be cast into our framework. For each candidate algorithm we learn good choices for parameters using a training set consisting of patches from a multi-image 3D reconstruction where accurate ground-truth matches are known. The best descriptors were those with log polar histogramming regions and feature vectors constructed from rectified outputs of steerable quadrature filters. At a 95% detection rate these gave one third of the incorrect matches produced by SIFT.

1. Introduction

Interest point detectors and descriptors have become popular for obtaining image to image correspondence for 3D reconstruction [17, 14], searching databases of photographs [10] and as a first stage in object or place recognition [8, 13]. In a typical scenario, an interest point detector is used to select matchable points in an image and a descriptor is used to characterize the region around each interest point. The output of a descriptor algorithm is a short vector of numbers which is invariant to common image transformations and can be compared with other descriptors in a database to obtain matches according to some distance metric. Many such matches can be used to bring images into correspondence or as part of a scheme for location recognition [16].

Various descriptor algorithms have been described in the literature [7, 1]. The SIFT algorithm is commonly used and has become a standard of comparison [9]. Descriptors are generally proposed *ad hoc* and there has been no systematic exploration of the space of algorithms. Local filters have been evaluated in the context of texture classification but not as region descriptors [15]. Carneiro and Jepson [4] evaluate their phase-based interest point descriptor against differential invariants by plotting ROC curves. Mikolajczyk and Schmid [11] have systematically compared the performance of ten recent descriptors and they advocate their GLOH descriptor which was found to outperform other candidates.

Descriptors algorithms typically contain a number of parameters which have so far required hand tuning. These parameters include smoothing factors, descriptor footprint size, number of orientation bins, etc. In [9] Lowe plots graphs in an attempt to manually optimize performance as parameters are varied. Since this approach is time consuming and unrealistic when a large number of parameters are involved, we attempt to automate the tuning process.

In [4] artificial image transformations were used to obtain ground truth matches. Mikolajczyk and Schmid [11] used natural images which were nearly planar and used camera motions which could be approximated as homographies. Ground truth homographies were then obtained semi-automatically. One disadvantage of these evaluation approaches is that 3D effects are avoided. We would like to evaluate descriptor performance when there can be non-planar motions around interest points and in particular with illumination changes and distortions typical of 3D viewing. We therefore use correspondences from reconstructed 3D scenes.

2. Our Contribution

There are three main contributions in this paper:

1. We generate a ground truth data set for testing and optimizing descriptor performance for which we have accurate match and non-match information. In particular, this data set includes 3D appearance variation around each interest point because it makes use of multiple images of a 3D scene where the camera matrices and 3D point correspondences are accurately recovered. This goes beyond planar-based evaluations.

2. Rather than testing *ad hoc* approaches, we break up the descriptor extraction process into a number of modules and put these together in different combinations. Cer-

1-4244-1180-7/07/$25.00 ©2007 IEEE

tain of these combinations give rise to published descriptors but many are untested. This allows us to examine each building block in detail and obtain a better covering of the space of possible algorithms.

3. We use learning to optimize the choice of parameters for each candidate descriptor algorithm. This contrasts with current attempts to hand tune descriptor parameters and helps to put each algorithm on the same footing so that we can obtain its best performance.

3. Obtaining 3D Ground Truth Data

The input to our descriptor algorithm is a square image patch while the output is a vector of numbers. This vector is intended to be descriptive of the image patch such that comparing descriptors should allow us to determine whether two patches are views of the same 3D point.

3.1. Generating Matching Patch Pairs

In order to evaluate our algorithms we obtained known matching and non-matching image patch pairs centered on virtual interest points by using the following approach:

We obtained a mixed training set consisting of tourist photographs of the Trevi Fountain and of Yosemite Valley (920 images), and a test set consisting of images of Notre Dame (500 images). We extracted interest points and matched them between all of the images within a set using the SIFT detector and descriptor [9]. We culled candidate matches using a symmetry criterion and used RANSAC [5] to estimate initial fundamental matrices between image pairs. This stage was followed by bundle adjustment to reconstruct 3D points and to obtain accurate camera matrices for each source image. A similar technique has been described by [17].

Once we had recovered robust 3D points that were seen from multiple cameras, these were projected back into the images in which they were matched to produce accurate virtual interest points. We then sampled 64×64 pixels around each virtual interest point to act as input patches for our descriptor algorithms. To define a consistent scale and orientation for each point we projected a virtual reference point, slightly offset from the original 3D point, into each image. The sampling scale was chosen to be 1 sample per pixel in the coarsest view of that point i.e. as high frequency as possible without oversampling in any image. The other images were sampled at the appropriate level of a scale-space pyramid to prevent aliasing.

Note that the patches obtained in this way correspond roughly to those used by the original SIFT descriptors that were matched. However, the position of the interest points is slightly altered by the bundle adjustment process, and the scales and orientations are defined in a different manner. Also, many of the correspondences identified may not have been matched using SIFT descriptors, but instead arose from the transitive closure of these matches across multiple source images.

Figure 1. Typical patches from our Trevi Fountain data set. Matching tiles are ordered consecutively.

For each 3D point we obtained numerous matching image patches. Large occlusions were avoided by projecting the 3D points only into images where the original SIFT descriptors had provided a match. However, in general there was some local occlusion present in the data set due to parallax and this was viewed as an advantage.

Our approach to obtaining ground truth matches can be compared with that of Moreels and Perona [12] who used 3D constraints across triplets of calibrated images to validate interest points as being views of the same 3D point.

From the multi-way matches in the Trevi Fountain and Yosemite Valley data set, we randomly chose 10,000 match pairs and 10,000 non-match pairs of 64×64 patches to act as a training set. For testing, we used the Notre Dame reconstruction and randomly chose 50,000 match pairs and 50,000 non-match pairs. Examples from the training set are shown in Figure 1. These data sets are now available on line[1].

3.2. Incorporating Jitter Statistics

Since our evaluation data sets were obtained by projecting 3D points into 2D images we expected that the normalization for scale and orientation and the accuracy of spatial correspondence of our patches would be much greater than that obtained directly from raw interest point detections. We therefore obtained statistics for the estimation of scale, orientation and position for interest points detected using the Harris-Laplace and SIFT DoG detectors [10, 9]. Our test set consisted of images from the Oxford Graffiti data set [2] and we applied 100 random synthetic affine warps to each with 1 unit of additive Gaussian noise. We detected interest points and estimated their sub-pixel location, scale and local orientation frame and we were able to histogram their errors between reference and warped images.[3]

In carrying out these experiments, we found that the the SIFT DoG detector produces better localization in scale and

[1] http://research.microsoft.com/ivm/PatchDataDownload

[2] http://www.robots.ox.ac.uk/ vgg/research/affine/index.html

[3] Histogram plots are available in the supplementary information.

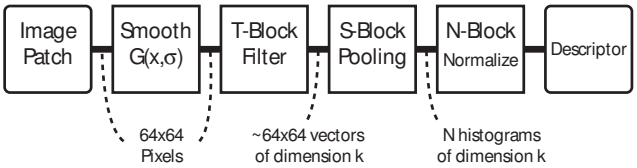

Figure 2. Processing stages in our generic descriptor algorithm.

position than the Harris-Laplace detector. Lowe's orientation histogramming technique, while seemingly more robust, gave only a marginally narrower error histogram than using a single local gradient and had more outliers.

Although we could not easily estimate them, we think that the *jitter* errors in our test set were much less than those present in real detection statistics. We therefore introduced random jitter by artificially warping the patches of our test set during descriptor evaluation to match the statistics obtained above. Introducing controlled amounts of jitter in this way allows us to learn descriptors which are robust to these effects.

4. Building a Descriptor Algorithm

Our generic descriptor algorithm consists of four processing stages (Figure 2). The input to the descriptor is a 64×64 image patch and the final output is vector of $D = kN$ numbers. We now describe each stage in turn, together with their candidate algorithms. Each candidate may be swapped in and out to produce a different overall descriptor. In addition, some candidates have a free parameters that we learn in order to maximize the performance of the descriptor as a whole.

4.1. Pre-smoothing

We smooth the image pixels using a Gaussian kernel of standard deviation σ_s as a pre-processing stage.

4.2. Transformation (T-block)

The transformation block maps the smoothed input patch onto a grid with one length k vector with positive elements per output sample. In this paper, the output grid was given the same resolution as the input patch. Various forms of linear or non-linear transformations or classifiers are possible. We experiment with the following:

[T1] We evaluate the gradient vector at each sample and recover its magnitude m and orientation θ. We then quantize the orientation to k directions and construct a vector of length k such that m is linearly allocated to the two circularly adjacent vector elements i and $i+1$ representing $\theta_i < \theta < \theta_{i+1}$ according to the proximity to these quantization centers. This process is equivalent to the orientation binning used in SIFT and GLOH.

[T2] We evaluate the gradient vector at each sample and rectify its x and y components to produce a vector of length 4: $\{|\nabla_x| - \nabla_x; |\nabla_x| + \nabla_x; |\nabla_y| - \nabla_y; |\nabla_y| + \nabla_y\}$. This provides a natural sine-weighted quantization of orientation into 4 directions. Alternatively we extend this to 8 directions by concatenating an additional length 4 vector using ∇_{45} which is the gradient vector rotated through $45°$.

[T3] We apply steerable filters at each sample location using n orientations and compute the magnitude response from quadrature pairs [6] to give a length $k = n$ vector. Alternatively we compute a length $k = 4n$ vector from the rectified quadrature pair responses directly in a similar way to the gradient computation described above. We tried two kinds of steerable filters: those based on a second derivatives provide broader orientation tuning while fourth order filters give narrow orientation tuning that can discriminate multiple orientations at each location in the input patch. These filters were implemented using the example coefficients given in [6].

[T4] We compute two isotropic Difference of Gaussians (DoG) responses with different center scales at each location by convolving the smoothed patch with three Gaussians (one additional center and two surrounds). The two linear DoG filter outputs are then used to generate a length 4 vector by rectifying their responses into positive and negative parts as described above for gradient vectors. We set the ratio between the center and surround space constants to 1.4. The pre-smoothing stage sets the size of the first DoG center and so we use one additional parameter to set the size of the second DoG center.

[T5] We compute the 4×4 Haar wavelet transform at overlapping locations and take the magnitude of each wavelet coefficient (minus the DC component) to give a length 15 vector at each sample position. Prior to computing the Haar transform, we independently bias and gain normalize each 4×4 patch to give a mean of zero and a standard deviation of one using $I'(\mathbf{x}) = (I(\mathbf{x}) - \mathrm{mean}(I))/\mathrm{stdev}(I)$.

[T6] We apply a fixed 4×4 classifier at overlapping sample locations. We previously learned the classifier by k-means clustering a large number of patches from natural images using 16 means. We independently bias and gain normalize each 4×4 patch for both classification and learning. During classification we pick the nearest mean to the 4×4 input patch and generate a length 16 vector having one non-zero element with value equal to unity.

[T7] We bias and gain normalize the smoothed input patch and then quantize the resulting signed gray value u to k levels. We generate a length k output vector at each sample point to represent the levels and linearly interpolate the value u into two adjacent elements i and $i+1$ representing $u_i < u < u_{i+1}$ such that the two elements have values $(u_{i+1} - u)/\delta$ and $(u - u_i)/\delta$ respectively. We set the quantization step size $\delta = u_{i+1} - u_i = 5/k$. This block was

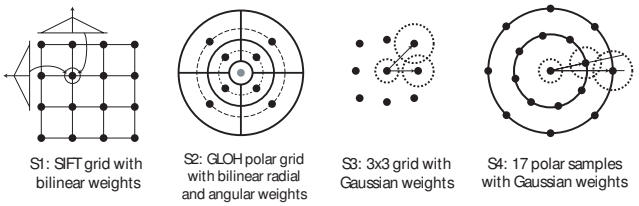

Figure 3. Examples of the different spatial summation blocks. For S3 and S4, the positions of the samples and the sizes of the Gaussian summation zones were appropriately parameterized.

inspired by Spin Images [7].

4.3. Spatial Pooling (S-block)

Many descriptor algorithms incorporate some form of histogramming. In our pooling stage we spatially accumulate weighted vectors from the previous stage to give N linearly summed vectors of length k and these are concatenated to form a descriptor of kN dimensions where $N \in \{3, 9, 16, 17, 25\}$. We now describe the different spatial arrangements of pooling and the different forms of weighting:

[S1] We used a square grid of pooling centers (see Figure 3) and learned the overall size of this grid. The vectors from the previous stage were summed together spatially by bilinearly weighting them according to their distance from the pooling centers as in the SIFT descriptor [9] so that the width of the bilinear function is dictated by the output sample spacing. We use sub-pixel interpolation throughout as this allows continuous control over the size of the descriptor grid. Note that all these operations are performed independently for each of the k vector elements.

[S2] We used the spatial histogramming scheme of the GLOH descriptor introduced by Mikolajczyk and Schmid [11]. This uses a polar arrangement of summing regions as shown in Figure 3. We used three variants of this arrangement with 3, 9 and 17 regions, depending on the number of angular segments in the outer two rings (zero, 4, or 8). The radii of the centers of the middle and outer regions and the outer edge of the outer region were parameters that were available for learning. Input vectors are bilinearly weighted in polar coordinates so that each vector contributes to multiple regions. As a last step, each of the final vectors from the N pooling regions is normalized by the area of its summation region.

[S3] We used normalized Gaussian weighting functions to sum input vectors over local pooling regions arranged on a 3×3, 4×4 or 5×5 grid. The sizes of each Gaussian and the positions of the grid samples were parameters that could be learned. Figure 3 displays the symmetric 3×3 arrangement with two position parameters and three Gaussian widths.

[S4] We tried the same approach as S3 but instead used a polar arrangement of Gaussian pooling regions with 17 or 25 sample centers. Parameters were used to specify the ring radii and the size of the Gaussian kernel associated with all samples in each ring (Figure 3). The rotational phase angle of the spatial positioning of middle ring samples was also a parameter that could be learned.

4.4. Post Normalization (N-block)

We use normalization to remove the descriptor dependency on image contrast. We employ the SIFT style normalization approach which involves range clipping descriptor elements. This algorithm consists of three steps: (1) Normalize to a unit vector, (2) clip all the elements of the vector that are above a threshold κ by computing $v_i' = \min(v_i, \kappa)$, and (3) re-normalize to a unit vector. This has the effect of reducing the dynamic range of the descriptor. The value κ was available for learning.

4.5. Reference Descriptors

We chose two descriptors to act as references for comparison purposes: the 128-dimensional SIFT descriptor and normalized sum squared differences (NSSD). For SIFT, there are three free parameters: the amount of Gaussian smoothing to apply to the patch before computing gradients (we used $\sigma = 2.7$ pixels), the width of the descriptor footprint (28.7 pixels), and the threshold to use during descriptor normalization ($\kappa = 0.154$). These values were obtained from our learning algorithm. For NSSD, we included pre-smoothing with $\sigma = 2.6$ pixels and a centered Gaussian windowing function with $\sigma = 24.3$ pixels also obtained from learning to optimize performance as described later.

5. Measuring Descriptor Performance

Once we have a candidate descriptor—constructed using a combination of the building blocks described above and with a specific choice of parameter values—we evaluate its performance over one of our data sets consisting of match and non-match pairs. For each pair we compute the Euclidean distance between descriptor vectors and form two histograms of this value for all true matching and non-matching cases in the data set. A good descriptor minimizes the amount of overlap of these histograms. We integrate the two histograms to obtain an ROC curve which plots correctly detected matches as a fraction of all true matches against incorrectly detected matches as a fraction of all true non-matches. We compute the area under the ROC curve as a good final score for descriptor performance and aim to maximize this value. Other choices for quality measures are possible depending on the application but we choose ROC area as a robust and fairly generic measure in lieu of inspecting the curves manually during optimization. In terms of reporting our results on the test set, however, we choose

to indicate performance in terms of the percentage of false matches present when 95% of all correct matches are detected.

Various researchers have made use of the ratio test when determining if two descriptors are a match or non-match [3]. In this approach the distance from one descriptor to its nearest neighbor in a database is compared to the distance to its second nearest neighbor and this ratio is used as a similarity measure. This technique produces a boost in performance at the cost of some complexity of implementation. In [11], Mikolajczyk and Schmid compare descriptors with and without the ratio test and find that it does not significantly change the order of descriptors ranking. We also chose not to use the ratio test because it would significantly complicate our process.

6. Learning Descriptor Parameters

Rather than using laborious hand-tuning to optimize parameters, we learned the most appropriate values by using Powell's multidimensional direction set method to maximize the ROC area. We initialized the optimization with reasonable choices of parameters. We also introduced jitter into the data set by pre-warping each patch using small random similarity warps with position, rotation and scale standard deviations of 0.4 pixels, 11 degrees and 0.12 octaves respectively.

Each ROC area measure was evaluated using one run over the training data set. After each run we updated the parameters and repeated the evaluation until the change in ROC area was small. Once we had determined optimal parameters, we re-ran the evaluation over our testing data set to obtain the final ROC curves and error rates.

7. Results for Interchanging T-blocks

Many combinations of computational blocks are possible. We chose to start our examination by studying the performance of the all T-blocks in combination with two different S-blocks. For all our results, we learned one parameter for smoothing, one for T4 (when present), a number of parameters for the various S-blocks and one for normalization.

We first study T-blocks when $k = 4$ and the results are shown in Figure 4, while selected ROC curves are in Figure 5. We first notice that the polar summation block, S2-17 (based on GLOH), always produces better results (lower error rates) than S1-16 (based on SIFT) and in some cases the error rate is nearly halved. We shall see later that this is due to log-polar summation and not due to the number of descriptor dimensions being larger for S2-17 or because it has more free parameters. From the ROC curves it can be seen the T1a, T2a and T4 in combination with S2-17 all perform better than SIFT (solid curve) with fewer descriptor dimensions (68 instead of 128). Particularly interesting

	S1-16	S2-17
T1a	Gradient, 4 orientations	
	6.51 (64)	3.92 (68)
T2a	Gradient, rectified into 4 bins	
	7.27 (64)	4.44 (68)
T3a	2nd order filter magnitude, 4 orientations	
	17.32 (64)	11.86 (68)
T3b	4th order filter magnitude, 4 orientations	
	17.80 (64)	14.50 (68)
T4	Rectified isotropic DoG filters	
	8.60 (64)	4.81 (68)
T7a	Intensity values quantized at 4 levels	
	15.77 (64)	10.88 (68)

Figure 4. Error rates (%) at 95% detection for four dimensional T-blocks ($k = 4$) in combination with S1-16 (SIFT-like) and S2-17 (GLOH-like) summation. Total descriptor dimensions are shown in brackets.

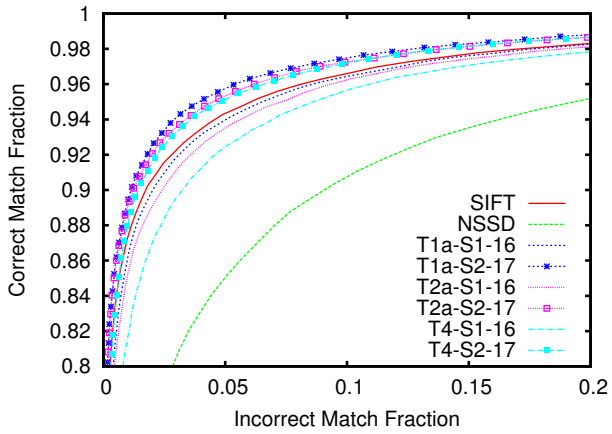

Figure 5. Selected ROC curves for the trained descriptors with four dimensional T-blocks ($k = 4$). Those that perform better than SIFT all make use of the S2 log-polar summation stage. See Table 4 for details.

is that the T2a block which involves simple rectification of the components of the gradient vector performs well with low computational burden.

The rectified DoG filter approach that we used in T4 gives a good result. In contrast with other T-blocks, this uses no orientation information but includes analysis at two different spatial frequency bands. The learning algorithm yields DoG center sizes which are separated by a factor of around four which minimizes the overlap of the filters in the frequency domain.

Surprisingly, second and fourth order steerable filters give poor results when the magnitude of quadrature responses is used in a feature vector, but, as we shall see shortly, if phase information is maintained then steerable filters are very effective.

Figures 6 and 7 show the performance of T-blocks when

	S1-16	S2-17
T1b	Gradient, 8 orientations	
	5.50 (128)	3.43 (136)
T2b	Gradient, rectified into 8 bins	
	6.74 (128)	3.71 (136)
T3c	2nd order filter magnitude, 8 orientations	
	16.91 (128)	11.74 (136)
T3d	4th order filter magnitude, 8 orientations	
	14.59 (128)	11.42 (136)
T7b	Intensity values quantized at 8 levels	
	16.95 (128)	13.95 (136)

Figure 6. Error rates (%) at 95% detection for eight dimensional T-blocks ($k = 8$) in combination with S1-16 (SIFT-like) and S2-17 (GLOH-like) summation. Total descriptor dimensions are shown in brackets.

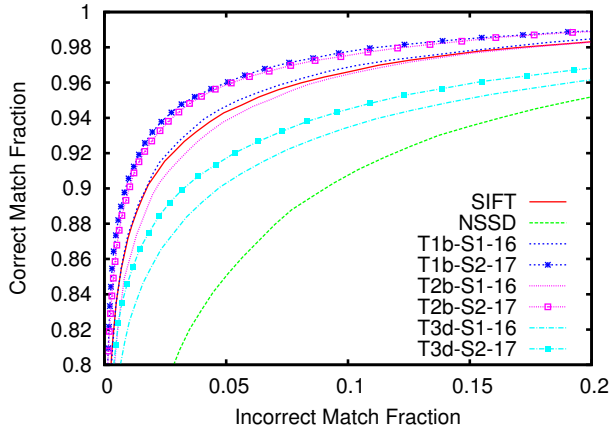

Figure 7. Selected ROC curves for the trained descriptors with eight dimensional T-blocks ($k = 8$). The simple gradient rectification of T2b gives good results. See Table 6 for details.

their output vector has eight dimensions. Increasing the number of feature dimensions reduces the error rate in all cases except for T7. In fact for T7, increasing the dimensions to 16 leads to even poorer performance. Since this block is based on image intensities, it is likely that more quantization levels leads to a greater dependence on inter-patch brightness variations despite the bias-gain normalization that we use.

Increasing the number of feature vector dimensions to $k = 16$ produces the results in Figures 8 and 9. There is once again an general increase in performance with, for example, T1c-S2-17 (2.98%) improving to half the error rate of SIFT (6.02%, Figure 13). Here we introduce a number of other T-blocks. The 4×4 classifier (T6) performs quite well. Haar wavelets on the other hand perform poorly. It may be that some other way of using them beyond taking the magnitude of coefficients could give different results.

The surprising additions here are the T3g and T3h blocks which, rather than combining the outputs of steerable filter quadrature pairs by taking the magnitude, instead keep them separate by rectification into different vector elements for positive and negative responses. This option on the T3 block produces extremely low error rates of around 2–3%. This is despite having the same feature dimensionality and a reduced number of orientation channels compared with T3e and T3f.

Mikolajczyk and Schmid [11] reported poor results for steerable filters, but it is important to note that here we are using them in a different way, as a feature element which is histogrammed, and not as a low dimensionality point descriptor.

	S1-16	S2-17
T1c	Gradient, 16 orientations	
	4.75 (256)	2.98 (272)
T3e	2nd order filter magnitude, 16 orientations	
	16.95 (256)	11.50 (272)
T3f	4th order filter magnitude, 16 orientations	
	13.29 (256)	11.07 (272)
T3g	2nd order quadrature filters, 4 orientations	
	5.68 (256)	2.96 (272)
T3h	4th order quadrature filters, 4 orientations	
	4.31 (256)	2.44 (272)
T5	4×4 Haar wavelets ($k = 15$)	
	44.49 (240)	38.35 (255)
T6	4×4 classifier	
	8.35 (256)	5.51 (272)
T7c	Intensity values quantized at 16 levels	
	18.25 (256)	15.25 (272)

Figure 8. Error rates (%) at 95% detection for sixteen dimensional T-blocks ($k = 16$) in combination with S1-16 (SIFT-like) and S2-17 (GLOH-like) summation. Total descriptor dimensions are shown in brackets.

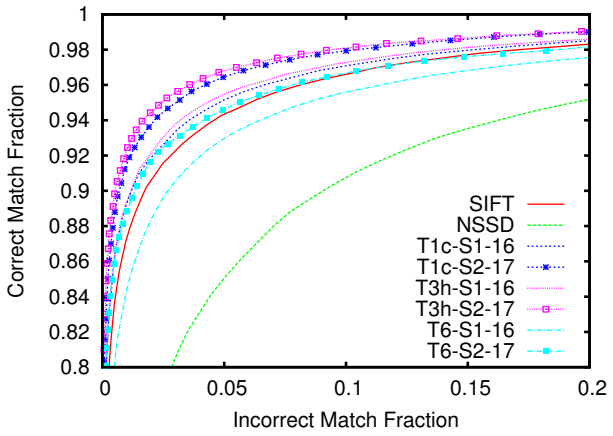

Figure 9. Selected ROC curves for the trained descriptors with 16 dimensional T-blocks ($k = 16$). The best result was obtained with T3h (fourth order quadrature steerable filters at four orientations). See Tables 8 and 13 for details. T3h-S4-24 is described later.

	T1b gradient, 8 orientations	
S1-9	SIFT 3×3 array	6.67 (72)
S1-16	SIFT 4×4 array	5.49 (128)
S1-25	SIFT 5×5 array	5.52 (200)
S2-3	GLOH, no angular segments	14.41 (24)
S2-9	GLOH, 4 angular segments	4.22 (72)
S2-17	GLOH, 8 angular segments	3.43 (136)
S3-9	3×3 Gaussian samples	5.46 (72)
S3-16	4×4 Gaussian samples	3.77 (128)
S3-25	5×5 Gaussian samples	3.46 (200)
S4-17	Center plus 2 rings of 8 samples	3.64 (136)
S4-25	Center plus 3 rings of 8 samples	3.20 (200)

Figure 10. Error rates (%) at 95% detection for different S-blocks in combination with the T1b block. Total descriptor dimensions are shown in brackets.

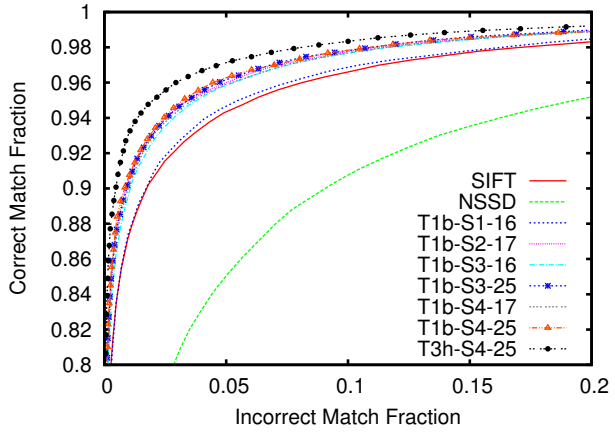

Figure 11. Selected ROC curves for the trained descriptors using 8 gradient orientations (T1b, $k = 8$) but with different S-blocks. The S2 (GLOH) footprint and all the highly parameterized S3 and S4 summation blocks gave around the same performance. See Table 10 for details.

All the descriptors with $k = 16$ and many others that perform quite well have a high dimensionality and this may be important to their performance. We have started preliminary investigations into using PCA to reduce the number of dimensions to give descriptor sizes which are more manageable for database search.

8. Results for Interchanging S-blocks

Next we experiment with different S-blocks while keeping the choice of T-block constant. We use the T1b gradient block with quantization to eight orientation bins as this generally performs quite well. Figure 10 shows the results. The main point here is that all the summation blocks (apart from S2-3) for which their parametrization allows for an approximately log-polar arrangement of histogram regions give about the same results and their performance is superior to the fixed rectangular lattice imposed by S1. This can be seen clearly from the convergence of the ROC curves (Figure 11). We find that beyond a certain point, increasing the number of samples N in the S-block does not result in much reduction in error rate.

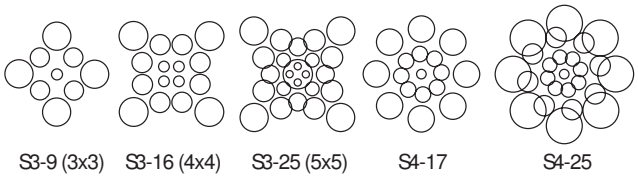

Figure 12. Examples of learned configurations of summation regions for the S3 and S4 blocks. These diagrams are to scale and circles represent the size of the Gaussian weighting function at each sample. The lattice positions and Gaussian sizes were initialized to be uniform. From left to right, the number of free parameters was 5, 7, 12, 6, and 9.

SIFT	Reference	6.02 (128)
NSSD	Reference	19.39 (4096)
T1b-S1-16	SIFT normalization	5.50 (128)
T1b-S1-16	Unit normalization	9.68 (128)
T3i-S1-16	2nd ord. quad. filter, 8 or.	6.55 (512)
T3i-S2-17	2nd ord. quad. filter, 8 or.	3.08 (544)
T3j-S1-16	4th ord. quad. filter, 8 or.	4.07 (512)
T3j-S2-17	4th ord. quad. filter, 8 or.	2.51 (544)
T3h-S4-25	4th ord. quad. filter, 4 or.	1.99 (400)

Figure 13. Error rates (%) at 95% detection for a variety of descriptors. Total descriptor dimensions are shown in brackets.

Even though S3 was originally designed to be a rectangular lattice, we used a number of parameters to set the positions of the samples and the learning algorithm was able to adjust these sample points into a better arrangement. This is clearly shown in Figure 12 which details the configuration of the S3 and S4 blocks after learning. The Gaussian weighted point samples are adjusted by the algorithm to conform to a more log-polar shape and the sizes of the Gaussians increase with distance from the descriptor center. In addition, the overlap between summation regions is kept to a minimum. These results provide strong support for the geometric blur approach of Berg *et al.* [2].

9. Additional Results

Figure 13 shows some additional results. We experimented with turning off the thresholding during the post normalization N-block. This had the effect of doubling the error rate, and from inspection of the changing parameters during the learning process it is apparent that a significant boost is always obtained by judicious choice of the N-block threshold parameter.

We tried increasing the number of orientation channels for the T3 block with quadrature filters to eight $k = 32$.

The results were similar or perhaps slightly worse than the situation with $k = 16$ (Figure 8) and at the cost of greatly increasing the number of descriptor dimensions.

The best result of all was obtained by combining steerable filters with the polar plan of S4 to give T3h-S4-25. At just under a 2% error rate, this is one third of the error rate produced by SIFT at 95% correct matches. The ROC curve for this descriptor is plotted on Figure 11. However the dimensionality is quite high at 400.

10. Discussion

We found that providing phase information is maintained in the feature vector, steerable filters worked remarkably well as local features along with log-polar summation. Fourth order filters gave better performance than second order filters. This may relate to their sharper orientation tuning which could potentially lead to less correlation between vector elements.

A number of questions remain that we wish to address in future work: Since our initial patches were extracted around SIFT DoG interest points, we might expect that the statistics of the patches could bias our learning to give best results with only the SIFT DoG detector. We aim to evaluate this by incorporating our descriptors into a matching system for 3D reconstruction where detectors can be interchanged. Another question concerns the relationship between patch scale and detected interest point scale and we have not yet addressed this although preliminary results show that good results are obtained over a variety of scale ratios. Finally we aim to carry out experiments with dimensionality reduction to see if we can maintain good performance from the better descriptors at reduced dimensions.

11. Conclusions

In conclusion, we have obtained a ground truth data set of 3D matching patches that can be used to evaluate prospective interest point descriptors. We have employed automated learning as a method of parameter adjustment and we have compared a number of candidate descriptors and identified some with a performance that exceeds the state of the art.

Acknowledgments

We would like to thank Sumit Basu for helping with the initial formulation of this problem in a learning framework and Larry Zitnick for his suggestion of the T6 block.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

[3] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, 2005.

[4] G. Carneiro and A. D. Jepson. Multi-scale phase-based local features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[5] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[6] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[7] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 319–324, 2003.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings IEEE International Conference on Computer Vision*, 1999.

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

[10] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings IEEE International Conference on Computer Vision*, 2001.

[11] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005.

[12] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. In *Proceedings IEEE International Conference on Computer Vision*, pages 800–807, 2005.

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

[14] M. Pollefeys and L. van Gool. From images to 3d models. *Communications of the ACM*, 45:50–55, 2002.

[15] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:291–310, 1999.

[16] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 226–231, 2002.

[17] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM Transactions on Graphics (SIGGRAPH Proceedings)*, volume 25, pages 835–846, 2006.