PCA-SIFT: A More Distinctive Representation for Local Image Descriptors

Yan Ke[1], Rahul Sukthankar[2,1]
{yke,rahuls}@cs.cmu.edu
[1] School of Computer Science, Carnegie Mellon University; [2] Intel Research Pittsburgh
http://www.cs.cmu.edu/˜yke/pcasift/

Abstract

Stable local feature detection and representation is a fundamental component of many image registration and object recognition algorithms. Mikolajczyk and Schmid [14] recently evaluated a variety of approaches and identified the SIFT [11] algorithm as being the most resistant to common image deformations. This paper examines (and improves upon) the local image descriptor used by SIFT. Like SIFT, our descriptors encode the salient aspects of the image gradient in the feature point's neighborhood; however, instead of using SIFT's smoothed weighted histograms, we apply Principal Components Analysis (PCA) to the normalized gradient patch. Our experiments demonstrate that the PCA-based local descriptors are more distinctive, more robust to image deformations, and more compact than the standard SIFT representation. We also present results showing that using these descriptors in an image retrieval application results in increased accuracy and faster matching.

1. Introduction

Local descriptors [6, 12, 18] are commonly employed in a number of real-world applications such as object recognition [3, 11] and image retrieval [13] because they can be computed efficiently, are resistant to partial occlusion, and are relatively insensitive to changes in viewpoint. There are two considerations to using local descriptors in these applications. First, we must localize the interest point in position and scale. Typically, interest points are placed at local peaks in a scale-space search, and filtered to preserve only those that are likely to remain stable over transformations. Second, we must build a description of the interest point; ideally, this description should be distinctive (reliably differentiating one interest point from others), concise, and invariant over transformations caused by changes in camera pose and lighting. While the localization and description aspects of interest point algorithms are often designed together, the solutions to these two problems are independent [14]. This paper focuses on approaches to the second aspect – the construction and evaluation of local descriptor representations.

Mikolajczyk and Schmid [14] presented a comparative study of several local descriptors including steerable filters [4], differential invariants [9], moment invariants [18], complex filters [16], SIFT [11], and cross-correlation of different types of interest points [6, 13]. Their experiments showed that the ranking of accuracy for the different algorithms was relatively insensitive to the method employed to find interest points in the image but was dependent on the representation used to model the image patch around the interest point. Since their best matching results were obtained using the SIFT descriptor, this paper focuses on that algorithm and explores alternatives to its local descriptor representation.

The remainder of this paper is organized as follows. Section 2 reviews the relevant aspects of the SIFT algorithm. Section 3 details our PCA-based representation for local features (PCA-SIFT). Section 4 presents our evaluation methodology and performance metrics. Section 5 provides detailed experimental results comparing PCA-SIFT to standard SIFT on feature-matching experiments and also in the context of an image retrieval application. Section 6 examines the reasons behind PCA-SIFT's accuracy by exploring the role of different components in the representation. Finally, Section 7 summarizes the contributions of this paper.

2. Review of the SIFT Algorithm

SIFT, as described in [12], consists of four major stages: (1) scale-space peak selection; (2) keypoint localization; (3) orientation assignment; (4) keypoint descriptor. In the first stage, potential interest points are identified by scanning the image over location and scale. This is implemented efficiently by constructing a Gaussian pyramid and searching for local peaks (termed keypoints) in a series of difference-of-Gaussian (DoG) images. In the second stage, candidate keypoints are localized to sub-pixel accuracy and eliminated if found to be unstable. The third identifies the dominant orientations for each keypoint based on its local image patch. The assigned orientation(s), scale and location for each keypoint enables SIFT to construct a canonical view for the keypoint that is invariant to similarity transforms. The final stage builds a *local image descriptor* for each keypoint, based upon the image gradients in its local neighborhood (discussed below in greater detail). The first three stages will not be discussed further in this paper since our work makes no contributions to those areas.

The final (keypoint descriptor) stage of the SIFT algorithm builds a representation for each keypoint based on a patch of pixels in its local neighborhood. Note that the patch has been previously centered about the keypoint's location, rotated on the basis of its dominant orientation and scaled to the appropriate size. The goal is to create a descriptor for the patch that is compact, highly distinctive (*i.e.,* patches from different keypoints map to different representations) and yet robust to changes in illumination and camera viewpoint (*i.e.,* the same keypoint in different images maps to similar representations). As discussed in [12], obvious approaches such as normalized correlation between image patches do not work since they are overly sensitive to registration errors and non-rigid deformations. The standard keypoint descriptor used by SIFT is created by sampling the magnitudes and orientations of the image gradient in the patch around the keypoint, and building smoothed orientation histograms to capture the important aspects of the patch. A 4×4 array of histograms, each with 8 orientation bins, captures the rough spatial structure of the patch. This 128-element vector is then normalized to unit length and thresholded to remove elements with small values.

The standard SIFT keypoint descriptor representation is noteworthy in several respects: (1) the representation is carefully designed to avoid problems due to boundary effects — smooth changes in location, orientation and scale do not cause radical changes in the feature vector; (2) it is fairly compact, expressing the patch of pixels using a 128 element vector; (3) while not explicitly invariant to affine transformations, the representation is surprisingly resilient to deformations such as those caused by perspective effects. These characteristics are evidenced in excellent matching performance against competing algorithms [14].

On the other hand, the construction of the standard SIFT feature vector is complicated and the choices behind its specific design (as given in [12]) are not clear. Our initial goal was to explore simpler alternatives and to empirically evaluate the tradeoffs. However, as discussed in the remainder of this paper, we discovered that our alternate representation was theoretically simpler, more compact, faster and more accurate than the standard SIFT descriptor. To ensure that our results are an accurate reflection of reality, we use the original SIFT source code and restrict our changes to the fourth stage.

3. PCA-based SIFT descriptors

Our algorithm for local descriptors (termed PCA-SIFT) accepts the same input as the standard SIFT descriptor: the sub-pixel location, scale, and dominant orientations of the keypoint. We extract a 41×41 patch at the given scale, centered over the keypoint, and rotated to align its dominant orientation to a canonical direction.[1] PCA-SIFT can be summarized in the following steps: (1) pre-compute an eigenspace to express the gradient images of local patches; (2) given a patch, compute its local image gradient; (3) project the gradient image vector using the eigenspace to derive a compact feature vector. This feature vector is significantly smaller than the standard SIFT feature vector, and can be used with the same matching algorithms. The Euclidean distance between two feature vectors is used to determine whether the two vectors correspond to the same keypoint in different images.

Principal Component Analysis (PCA) [7] is a standard technique for dimensionality reduction and has been applied to a broad class of computer vision problems, including feature selection (*e.g.,* [5]), object recognition (*e.g.,* [15]) and face recognition (*e.g.,* [17]). While PCA suffers from a number of shortcomings [8, 10], such as its implicit assumption of Gaussian distributions and its restriction to orthogonal linear combinations, it remains popular due to its simplicity. The idea of applying PCA to image patches is not novel (*e.g.,* [3]). Our contribution lies in rigorously demonstrating that PCA is well-suited to representing keypoint patches (once they have been transformed into a canonical scale, position and orientation), and that this representation significantly improves SIFT's matching performance. PCA-SIFT is detailed in the following subsections.

3.1. Offline computation of patch eigenspace

PCA enables us to linearly-project high-dimensional samples onto a low-dimensional feature space. For our application, this projection (encoded by the patch eigenspace) can be pre-computed once and stored.

As discussed above, the input vector is created by concatenating the horizontal and vertical gradient maps for the 41×41 patch centered at the keypoint. Thus, the input vector has $2 \times 39 \times 39 = 3042$ elements. We then normalize this vector to unit magnitude to minimize the impact of variations in illumination. It is important to note that the 41×41 patch does not span the entire space of pixel values, nor the smaller manifold of patches drawn from natural images; it consists of the highly-restricted set of patches that passed through the first three stages of SIFT. More precisely, each of the patches satisfies the following properties: (1) it is centered on a local maximum in scale-space; (2) it has been rotated so that (one of its) dominant gradient orientations is aligned to be vertical; (3) it only contains information for the scale appropriate to this keypoint – *i.e.,* the 41×41 patch

[1] For keypoints with multiple dominant orientations, we build a representation for each orientation, in the same manner as SIFT.

may have been created from a much larger region from the original image. The remaining variations in the input vector are mainly due to the “identity” of the keypoint (*i.e.*, the 3-D scene corresponding to this location) or to unmodeled distortions (such as perspective effects caused by changing camera viewpoint). It is not unreasonable to believe that these remaining variations can be reasonably modeled by low-dimensional Gaussian distributions, enabling PCA to accurately represent them with a compact feature representation. More importantly, projecting the gradient patch onto the low-dimensional space appears to retain the identity-related variation while discarding the distortions induced by other effects. This hypothesis is supported by the experimental evidence discussed in Sections 4 and 6.

To build our eigenspace, we ran the first three stages of the SIFT algorithm on a diverse collection of images and collected 21,000 patches. Each was processed as described above to create a 3042-element vector, and PCA was applied to the covariance matrix of these vectors. The matrix consisting of the top n eigenvectors was stored on disk and used as the projection matrix for PCA-SIFT. The images used in building the eigenspace were discarded and not used in any of the matching experiments.

3.2. Feature representation

To find the feature vector for a given image patch, we simply create its 3042-element normalized image gradient vector and project it into our feature space using the stored eigenspace. We empirically determined good values for the dimensionality of the feature space, n; most of the results described in this paper use $n = 20$ (Section 6 discusses the effects of n on performance). The standard SIFT representation employs 128-element vectors; using PCA-SIFT results in significant space benefits.

As discussed above, we use the Euclidean distance between two feature vectors to determine whether the two vectors belong to the same keypoint in different images. Thresholding this distance generates a binary decision, and adjusting this threshold enables one to select the appropriate trade-off between false positives and false negatives.

4. Evaluation

First, we discuss the evaluation metrics used to quantify our results. We then outline our experimental setup and discuss the issue of generating ground-truth data. Results are presented in Section 5.

4.1. Evaluation metrics

Receiver Operating Characteristics (ROC) and Recall-Precision are both popular metrics in the literature, and are sometimes used interchangeably. Both capture the fact that we want to increase the number of correct positives while minimizing the number of false positives; however, the subtle differences between the metrics should dictate the choice of one over the other for specific scenarios. As observed in [2], the former is well-suited for evaluating *classifiers*, since the false detection rate is well-defined; the latter is better-suited for evaluating *detectors*, since the number of false detections *relative to the total number of detections* is correctly expressed by $1-precision$ even though the total number of negatives cannot be determined.

Following [14], we chose to measure the performance of the SIFT local descriptor representations on a keypoint matching problem, defined as follows: given an interest point in one image, find all matches of that interest point in the dataset. Clearly, this is a *detection* rather than a *classification* task (the total number of negatives is not well-defined), and thus the appropriate metric is recall-precision. Therefore, although [14] uses ROC curves, this paper presents *recall* vs. $1-precision$ graphs.

These graphs are generated as follows. The keypoints for all of the images in the dataset are identified (using the initial stages of the SIFT algorithm). All pairs of keypoints from different images are examined. If the Euclidean distance between the feature vectors for a particular pair of keypoints falls below the chosen threshold, this pair is termed a *match*. A *correct-positive* is a match where the two keypoints correspond to the same physical location (as determined either by ground truth for labeled images, or using known image transformations for synthetic image deformation tests). A *false-positive* is a match where the two keypoints come from different physical locations. The *total number of positives* for the given dataset is known *a priori*. From these numbers, we can determine *recall* and $1-$precision:

$$recall = \frac{\text{number of } \textit{correct-positives}}{\text{total number of positives}}$$

and

$$1-precision = \frac{\text{number of } \textit{false-positives}}{\text{total number of matches (correct or false)}}.$$

We generate the *recall* vs. $1-precision$ graphs for our experiments by varying the threshold for each algorithm.

4.2. Experimental setup

We ran three main types of experiments to explore the difference between the standard SIFT representation and PCA-SIFT. The first type examined each descriptor’s robustness to (synthetically-generated) effects caused by the addition of noise, changes in illumination and the application of image transformations. We collected a dataset of images[2] and applied the following transformations to each

[2] Available at: `http://www.cs.cmu.edu/~yke/pcasift/`

image: (1) Gaussian noise (σ=0.05), where image intensities range from 0 to 1; (2) rotation of 45° followed by a 50% scaling; (3) intensity scaling of 50%; (4) projective warp equivalent to a viewing angle change of approximately 30°.

Three descriptors were evaluated in these experiments: (1) the standard SIFT feature representation (denoted as “SIFT”); (2) PCA-SIFT (n=20) as described in Section 3, where the eigenspace models the local gradient (denoted as “Grad PCA 20”); (3) a variant of PCA-SIFT where the eigenspace directly models the local image patch rather than the local gradient (denoted as “Img PCA 20”). For (3), we employed the standard intensity normalization technique of subtracting the mean and scaling to unit variance; without this step, the results for (3) would be even worse.

The second type evaluated both descriptors on real images taken from varying viewpoints, such as the INRIA MOVI Graffiti datasets [1]. The third type involved integrating SIFT and PCA-SIFT into an image retrieval application. Additional experiments to investigate the effect of keypoint localization error, number of PCA components, choice of distance metric are given in Section 6.

4.3. Generation of ground truth data

We run the initial stages of SIFT on every image to identify the keypoints. The goal is to obtain, for every pair of images, a list of the correct keypoint matches. Since the SIFT algorithm generates hundreds or thousands of keypoints per image, manually identifying matches for a large dataset would be extremely time-consuming and potentially error-prone. Fortunately, knowing the mapping between two images enables us to automatically resolve the matching problem. For experiments involving synthetically-generated target images, these image transforms are known *a priori*. For the experiments involving the Graffiti dataset, the transform (expressed as a homography) between two matching scenes is given. We use that information as follows.

Let T_{ij} be the transformation mapping a point in image i to its corresponding point in image j. An interest point in the first image, P_i, maps to $P'_i = T_{ij}(P_i)$ in the second image. Ideally, one should expect a matching interest point in the second image to lie at $P_j = P'_i$. In practice, we consider the match to be valid if P_j and P'_i are sufficiently close in space and scale. Two points are considered sufficiently close in space if the distance between them is less than σ pixels, where σ is the standard deviation of the Gaussian used to generate the DoG function. They are considered sufficiently close in scale if their scales are within $\sqrt{2}$ of each other.

5. Results

This section presents results comparing PCA-SIFT to the standard SIFT representation on controlled experiments, the Graffiti dataset, and on an image retrieval task.

5.1. Controlled transformations

Figure 1 presents the results of the first set of matching experiments, where images were distorted under controlled conditions.

Figure 1a shows that PCA-SIFT is dramatically better at handling noisy images for almost all values of $1-$precision, and that PCA-SIFT (on the gradient map) dominates the PCA representation of the local image patch. The standard SIFT representation outperforms PCA-SIFT only when extremely high false positives rates can be tolerated. These results are not particularly surprising since PCA should provide excellent reconstruction under the effects of Gaussian noise. Our next experiments examine the impact of geometric transforms.

Figure 1b plots results of an experiment where target images were rotated by 45° and scaled by 50% while Figure 1c shows matches after targets were distorted with perspective transformation corresponding to a 30° out-of-plane rotation. While none of the representations are particularly well-suited to this task, PCA-SIFT clearly dominates the other two algorithms.

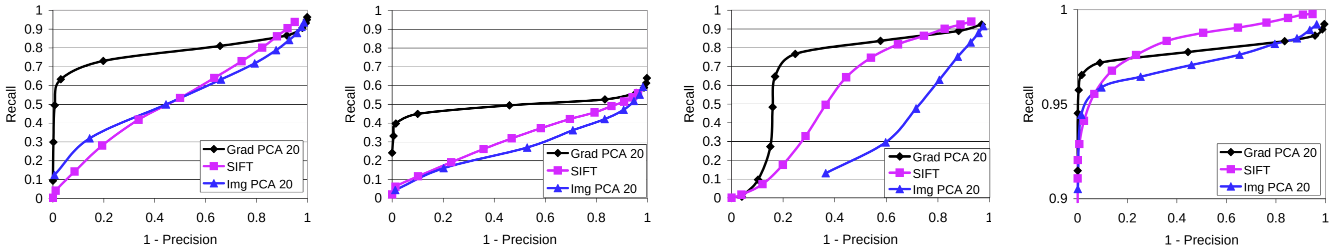
Figure 1d shows that all of the representations are well-suited to capturing simple variations in illumination (note that *recall* axis has been magnified and offset to highlight the differences). Looking closely, we can see that the standard SIFT representation is slightly better than PCA-SIFT for most of the domain. However, given that all of the algorithms display a recall of more than 95%, this is not very significant.

5.2. INRIA Graffiti dataset

The INRIA Graffiti [1] dataset contains images of graffiti-covered walls taken from different camera viewpoints. Since the scene is planar, the transformation between images can be modeled as a 2-D planar homography (given in the dataset). Our goal was to match corresponding keypoints between images. Figure 2 shows the matching performance for the two algorithms on the Graffiti 6 dataset. Although the absolute *recall* is rather low due to the high degree of projective warp, PCA-SIFT clearly dominates. Note that a low recall at high precision is acceptable for real-world applications. For instance, PCA-SIFT has a recall of about 5% at $1-$precision of 20%; we locate about a thousand keypoints in the image, of which 50 are reliable matches. This number of reliable matches is sufficient for applications such as image retrieval.

5.3. Image retrieval application

We have integrated SIFT and PCA-SIFT into an image retrieval application for real-world scenes taken from differ-



(a) add noise (b) rotate & scale (c) projective warp (d) reduce brightness

Figure 1: SIFT vs. PCA-SIFT on a matching task where the images are deformed or corrupted under controlled conditions. (a) target images were corrupted by Gaussian noise (σ=0.05 of the intensity range). (b) target images were rotated by 45° and scaled by 50%. (c) target images were projectively warped to simulate a viewpoint change of 30°. (d) target image intensities were scaled by 50%. Note that the Recall axis starts at 0.9 because all methods perform extremely well on this task.

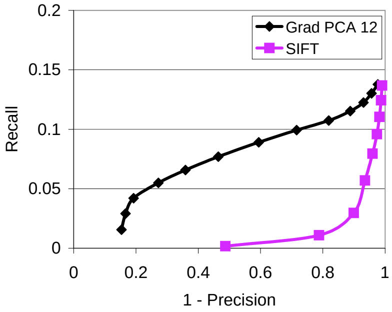


Figure 2: SIFT vs. PCA-SIFT (n=12) on Graffiti 6 [1].

ent viewpoints. Unlike the Graffiti dataset, the scenes are not planar, and contain occlusions, and reflective surfaces.

Image retrieval using SIFT is formulated as follows. Given two images, we first extract their corresponding feature vectors. For each feature vector in one image, we compare it against all feature vectors in the other image and count the number of features that are within a threshold distance. We treat the number of matches as a similarity between images.

For this experiment, we chose a small dataset[3] of 30 images (10 common household items, photographed from different viewpoints). Each image was used as a query into the database (in a leave-one-out manner). If both of the other two images of the corresponding object were returned in the top three positions, the algorithm was awarded 2 points; if only one of the correct matches appeared in the top three positions, it was awarded 1 point; otherwise, it was awarded no points. The scores were divided by 60 (the total number of correct matches) and are given in Table 1.[4] The threshold distance for each algorithm was tuned to give the best results (SIFT threshold: 141; PCA-SIFT threshold: 2200); in practice, a wide range of thresholds works well. The results show that PCA-SIFT's matching accuracy at the keypoint level also translates into better retrieval results.

	SIFT	PCA-SIFT (n=20)
Correct retrieval	43%	68%

Table 1: Percentage of images correctly retrieved in our image retrieval application. PCA-SIFT's matching accuracy translates into significant practical benefits.

Figure 3 shows the result of applying SIFT to two challenging scenes (a cluttered coffee table and a Graffiti image). We manually set the thresholds to have each algorithm return 10 matches. PCA-SIFT clearly dominates the standard representation in these experiments. In particular, the latter appears to get confused by the edges of several of the objects.

Figure 4 is a detailed look at one of the keypoints from this example. The potential matches are presented in rank order for each algorithm. The standard representation rates the correct match in the third position while PCA-SIFT correctly ranks it first.

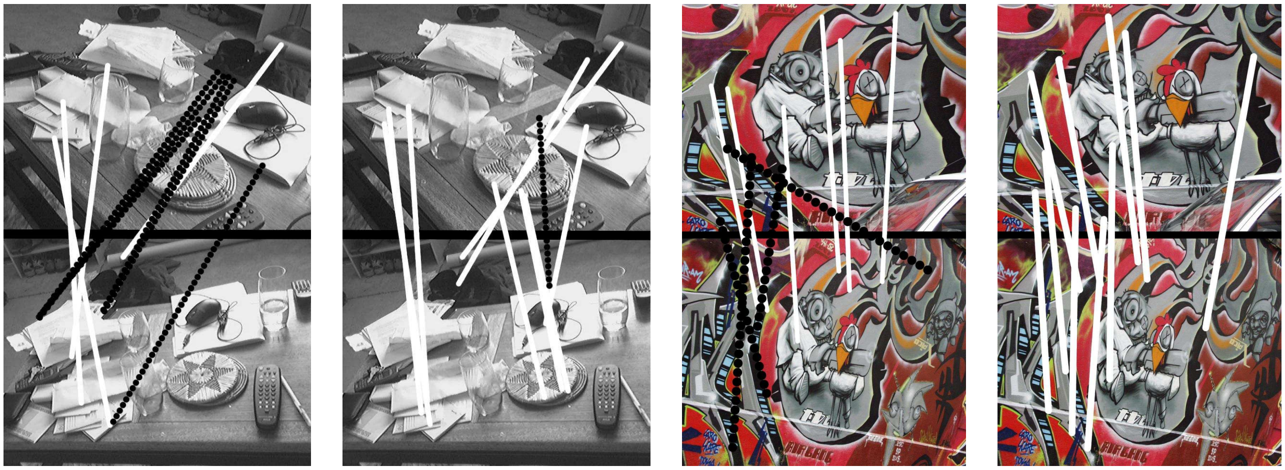
Table 2 compares the running time between SIFT and PCA-SIFT. The top part shows the feature extraction of an image with approximately 2200 interest points. The first row is the time needed to localize the interest point (common to both algorithms). The second and third rows show the time needed to calculate the descriptor representation. We observe that the time needed to compute the representation is comparable. The lower part of the table shows that PCA-SIFT is significantly faster in the matching phase; PCA-SIFT (n=20) requires only a third of the time to do the 2.4 million point comparisons.

6. Discussion

Section 5 showed that PCA-SIFT was both significantly more accurate and much faster than the standard SIFT local descriptor. However, these results are somewhat surprising since the latter was carefully designed while PCA-SIFT is a somewhat obvious idea. We now take a closer look at the algorithm, and propose some hypotheses that may help

[3] Available at `http://www.cs.cmu.edu/~yke/pcasift/`

[4] This is equivalent to measuring P(2), the *precision* of the system when two objects are retrieved.



(A1) SIFT: 4/10 correct (A2) PCA-SIFT (n=20): 9/10 correct (B1) SIFT: 6/10 correct (B2) PCA-SIFT (n=20): 10/10 correct

Figure 3: A comparison between SIFT and PCA-SIFT (n=20) on some challenging real-world images taken from different viewpoints. (A) is a photo of a cluttered coffee table; (B) is a wall covered in Graffiti from the INRIA Graffiti dataset. The top ten matches are shown for each algorithm: solid white lines denote correct matches while dotted black lines show incorrect ones.

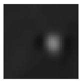
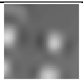
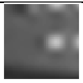
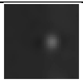
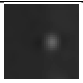
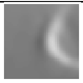
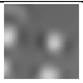
Query keypoint			
Rank	1	2	3
SIFT SIFT Distance PCA-SIFT Distance	 158 8087	 245 8551	 256 4438
PCA-SIFT SIFT Distance PCA-SIFT Distance	 256 4438	 399 7011	 158 8087

Figure 4: A closer look at matching results for a particular keypoint (zoomed in view of a region from Figure 3). The top three matches for this keypoint for SIFT and PCA-SIFT (n=20) are shown. The correct match is third on the list for the standard representation, while it is the top match for PCA-SIFT. The two algorithms use different feature spaces so a direct comparison of the distance values is not meaningful.

	time (sec)	σ
Localization and I/O	2.63	0.09
SIFT representation	1.59	0.06
PCA-SIFT representation	1.64	0.04
SIFT matching	2.20	0.03
PCA-SIFT matching	0.58	0.05

Table 2: Running times (and standard deviations) for SIFT and PCA-SIFT (n=20) averaged over 10 independent runs. The localization and I/O steps are common to both algorithms. Building both representations takes comparable time, but matching using PCA-SIFT is much faster.

explain PCA-SIFT's success.

Figure 5 shows that the gradient patches surrounding SIFT keypoints are highly structured, and therefore easier to represent using PCA. We see that the eigenvalues for these patches decay much faster than eigenvalues for randomly-selected gradient patches. This is because the patches surrounding keypoints all share certain characteristics, stemming from the SIFT keypoint detection phase: they are all centered on a local maximum in scale-space, rotated to align the dominant gradients with the vertical, and scaled appropriately. This simplifies the job that PCA must do, and enables PCA-SIFT to accurately represent the patch using a small number of dimensions.

A natural question to consider is whether PCA-SIFT's representation is sensitive to the images used in the creation of the eigenspace — *i.e.*, is it reasonable to assume a canonical eigenspace for representing keypoint gradient patches? Our experiments show that the eigenspaces generated from keypoint patches obtained from different image sets are quite consistent, and that PCA-SIFT's performance does not require us to train an eigenspace specifically for the testing images. Figure 6 shows the results of two controlled transformation experiments (described in Section 4.2) performed with two different eigenspaces. In the first experiment, the target images were rotated by 45°; in the second, the target images were projectively warped. The first eigenspace was tailored specifically to the images used in

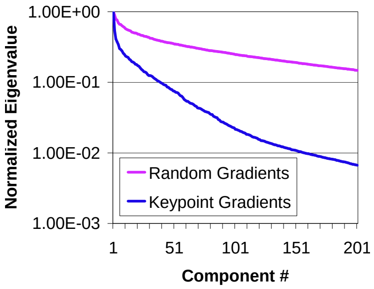
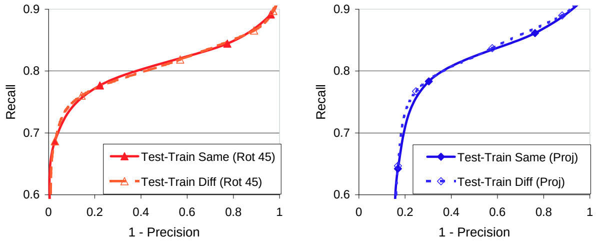


Figure 5: This graph shows that PCA on randomly-selected gradient image patches differs from PCA-SIFT, where PCA is applied to gradient patches surrounding SIFT keypoints. The eigenvalues for PCA-SIFT decay much more rapidly, supporting our belief that PCA-SIFT successfully represents keypoints using a small number of dimensions because PCA is only required to capture the appearance of a very particular type of image patch.

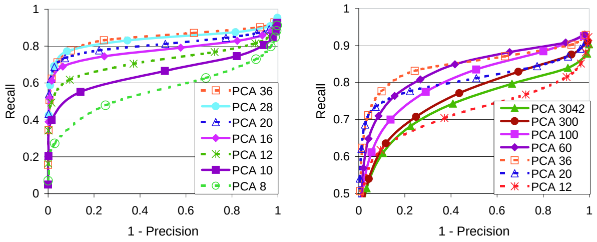


(a) Rotate 45° (b) Projective warp 30°

Figure 6: PCA-SIFT's performance is not sensitive to the images used in the creation of the eigenspace (graph magnified to highlight differences). The solid lines show results using an eigenspace that was trained on keypoint patches from the test set (the optimal training set) while the dashed lines show results using our standard eigenspace. Using a customized eigenspace makes almost no difference. This strongly supports our belief that the entire space of keypoint patches is well represented by a single eigenspace.

this test; the second eigenspace was trained on an unrelated set of images (and is the eigenspace used in all of our experiments). On each experiment, PCA-SIFT's performance with either eigenspace is almost identical; the differences are too small to be seen in an unscaled graph, requiring us to magnify Figure 6 to highlight them.

Figure 7a shows the relationship between PCA-SIFT's matching accuracy and the dimensionality of the feature space (for small values of n). As expected, increasing the dimensionality of the feature vector results in better accuracy, since the representation is able to capture the structure of the gradient patch with better fidelity. As we continue adding dimensions to the feature vector, the marginal benefit slows. Figure 7b shows the same graph for greater values of n, as the number of dimensions in the representation approaches the number of dimensions of the input vector. Now we see an interesting phenomenon: once n exceeds a certain size, the matching accuracy of the algorithm actually begins to decline. At n=3042, where PCA performs a perfect (loss-less) reconstruction of the input vector, the matching accuracy is only slightly better than n=12. Our hypothesis for this is that the first several components of the PCA subspace are sufficient for encoding the variations in the gradient patch caused by the identity of the keypoint, (which we would like to model accurately) while the later components represent detail in the gradient image that is not useful, or potentially detrimental, such as distortions from projective warps. There is also trade-off between running time and the dimensionality of the feature vector. Using fewer components requires less storage and results in faster matching. We obtain good results with n=20.

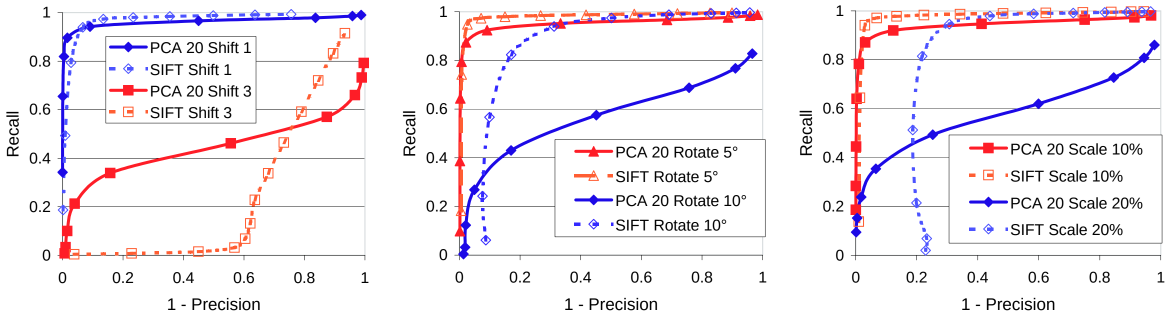


(a) (b)

Figure 7: PCA-SIFT performance as PCA dimension (n) is varied. $n = 36$ achieves the best matching performance.

One of the reasons cited for SIFT's matching success is that its feature representation has been carefully designed to be robust to localization error. By contrast, PCA is known to be notoriously sensitive to registration error. Figure 8a shows an experiment where we introduce registration error after the localization stage. We add 1 and 3 pixel errors (at the appropriate scale) in a random direction with respect to the dominant orientation. Somewhat surprisingly, PCA-SIFT continues to perform well. However, when error is introduced into the orientation assignment phase (Figure 8b) or in the scale estimation (Figure 8c), we see that PCA-SIFT's accuracy begins to degrade. This confirms our belief that the standard SIFT representation is better suited at handling these errors. However, our experiments show that the feature localization stage of the SIFT algorithm is very good at centering, orienting and scaling the keypoint patches; thus these benefits of the standard SIFT representation are rarely (if ever) observed in practice.

We hypothesize that PCA-SIFT's matching accuracy can be attributed to several factors. First, using the gradient patch rather than the raw patch around the keypoint makes the representation robust to illumination changes, and reduces the variation that PCA needs to model. Second, the pre-processing performed by the first three stages of SIFT simplifies the modeling problem for PCA since the remainder of the variation is due to keypoint identity and perspective distortions. Third, discarding the lower components in



(a) shift (b) rotation (c) scale

Figure 8: SIFT vs. PCA-SIFT (n=20) when additional error is directly introduced in the feature localization stage. The estimates given by SIFT's localization stage were shifted by $k\sigma$ pixel (k=1,3; σ is scale of the extracted feature). PCA is often observed to be sensitive to registration error; however, PCA-SIFT performs very well under these conditions. When the localization is corrupted by errors in rotation (5° and 10°) or scale (10% and 20%), SIFT outperforms PCA-SIFT as expected because SIFT is specifically engineered to be robust against detector error.

PCA improves accuracy by eliminating the variations due to unmodeled distortions. Finally, using a small number of dimensions provides significant benefits in storage space and matching speed.

7. Conclusion

This paper introduced an alternate representation for local image descriptors for the SIFT algorithm. Compared to the standard representation, PCA-SIFT is both more distinctive and more compact leading to significant improvements in matching accuracy (and speed) for both controlled and real-world conditions. We believe that, although PCA is ill-suited for representing the general class of image patches, it is very well-suited for capturing the variation in the gradient image of a keypoint that has been localized in scale, space and orientation. We are currently extending our representation to color images, and exploring ways to apply the ideas behind PCA-SIFT to other keypoint algorithms.

8. Acknowledgments

Yan Ke is supported by the Intel Research Scholar program. Thanks to David Lowe for the SIFT source code and helpful advice on this topic. Thanks also to Larry Huston, Phil Gibbons, M. Satyanarayanan, and Martial Hebert for valuable feedback.

References

[1] Viewpoint change sequences. `http://www.inrialpes.fr/movi/`.

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of European Conference on Computer Vision*, pages 113–130, 2002.

[3] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of Computer Vision and Pattern Recognition*, June 2003.

[4] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[5] K. Fukunaga and W. Koontz. Application of the Karhunen-Loeve expansion to feature selection and ordering. *IEEE Trans. Communications*, 19(4), 1970.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[7] I. T. Joliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[8] J. Karhunen and J. Joutsensalo. Generalization of principal component analysis, optimization problems and neural networks. *Neural Networks*, 8(4), 1995.

[9] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. In *Biological Cybernetics*, volume 55, pages 367–375, 1987.

[10] D. Lee and S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 1999.

[11] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of International Conference on Computer Vision*, pages 1150–1157, 1999.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[13] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of International Conference on Computer Vision*, pages 525–531, July 2001.

[14] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of Computer Vision and Pattern Recognition*, June 2003.

[15] H. Murase and S. Nayar. Detection of 3D objects in cluttered scenes using hierarchical eigenspace. *Pattern Recognition Letters*, 18(4), April 1997.

[16] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. In *Proceedings of European Conference on Computer Vision*, volume 1, pages 414–431. Springer-Verlag, 2002.

[17] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings of Computer Vision and Pattern Recognition*, 1991.

[18] L. Van Gool, T. Moons, and D. Ungureanu. Affine/photometric invariants for planar intensity patterns. In *Proceedings of European Conference on Computer Vision*, 1996.