Fast and Scalable Keypoint Recognition and Image Retrieval using Binary Codes

Jonathan Ventura
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
jventura@cs.ucsb.edu

Tobias Höllerer
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
holl@cs.ucsb.edu

Abstract

In this paper we report an evaluation of keypoint descriptor compression using as little as 16 bits to describe a single keypoint. We use spectral hashing to compress keypoint descriptors, and match them using the Hamming distance. By indexing the keypoints in a binary tree, we can quickly recognize keypoints with a very small database, and efficiently insert new keypoints. Our tests using image datasets with perspective distortion show the method to enable fast keypoint recognition and image retrieval with a small code size, and point towards potential applications for scalable visual SLAM on mobile phones.

1. Introduction

Historically, there have been two dominant approaches to visual keypoint recognition: a) compute a complex descriptor from a single observation that is robust to some geometric and photometric distortion; b) compute a simple descriptor that is not very robust by itself, but can be aggregated from many observations, e.g. synthetically generated ones. Examples of complex descriptors include the SIFT detector/descriptor [9] and the SURF algorithm inspired by it [2]. Examples of simple descriptors from many observations include the Ferns algorithm [11] and Histogrammed Intensity Patches (HIPs) [16].

Both approaches to keypoint recognition have had much success, but also have inherent disadvantages. Complex descriptors like SIFT and even SURF are difficult to use in real-time on constrained mobile platforms because of their time and memory requirements. Also, these descriptors are usually matched using Euclidean distance, with an indexing structure such as a kd-tree, which does not handle real-time insertion very well. This limits its usefulness when adding new keypoints in real-time. Approaches with a simple descriptor such as Ferns or HIPs are designed for fast detection and matching at run-time. However, generating the indexing structure is slow and memory-intensive, since it involves generating thousands of warps of the source image. This process does not scale well, and is not suitable for real-time insertion of new images.

In this paper we evaluate the application of a binary hashing method called spectral hashing [20] to complex keypoint descriptors. There are two significant advantages in using binary-hashed descriptors. First, they are much smaller than the original descriptor; we show that as little as 16 bits are sufficient for matching in some cases (Section 3). Second, we use Hamming distance for matching, which is extremely fast to compute, and can be sped up even more by using a simple indexing structure which handles fast insertion as well (Section 4). Our method can also be used to recognize multiple targets (Section 5). Our evaluation shows that matching of spectrally hashed descriptors performs nearly as well as sum of squared differences (SSD) matching, but is significantly faster and more space efficient (Section 6).

2. Related Work

The SIFT detector and descriptor designed by David Lowe [9] is a 128-dimensional descriptor (1024 bits) which first detects the orientation and scale of a keypoint, and histograms gradient values from sub-patches sampled around the keypoint. The SIFT descriptor has been quite successful in a wide variety of image-matching applications, but is generally considered an "offline" algorithm because of its computational requirements. The SURF detector and descriptor [2] adapts the concept of SIFT but decreases detection and descriptor computation time by using integral images, without significant loss of precision. These descriptors and similar ones are matched in Euclidean space, often with the aid of a space-partitioning structure such as a kd-tree. Although these methods exhibit robust performance, they incur a high computation cost, large database size, and little support for expansion of the feature database at run-time.

A different breed of keypoint recognition instead learns the appearance of features by generating thousands of warps of the input image. The Ferns approach [11] uses randomly chosen binary tests on the intensity values around a keypoint in a Naive Bayesian classifier. Taylor *et al*. described the Histogrammed Intensity Patch, which learns heavily quantized intensity distributions of warped keypoints [16]. With both of these cases, keypoint recognition at runtime is fast, since we only need to apply a set of binary tests to the input image to classify a keypoint. However, the need to generate warped images limits database expansion at runtime. Also, there has not been a demonstration of the robustness of these learned approaches when using a large database of tracking targets.

Some hybrid approaches exist which combine the robustness of complex descriptors with the speed and compactness of simpler descriptors. Wagner *et al*. studied how to best achieve natural feature tracking on a constrained mobile platform, by adjusting SIFT and Ferns to be more efficient [18, 19]. The BRIEF descriptor simply uses randomly chosen binary tests by themselves to form a bit-vector descriptor which demonstrates impressive performance [3]. Chandrasekhar et al. developed a 60-bit compressed histogram of gradients (CHoG) descriptor targeted towards minimal use of bandwidth when performing image retrieval queries over the network [5]. In our work we consider the performance of compressed descriptors as small as 16 bits, and allow for efficient real-time insertion of new images to the matching database.

This paper focuses on evaluating compression of a complex descriptor into a small bit vector for efficient keypoint recognition. Torralba, Fergus and Weiss demonstrated content-based image retrieval using binary hashing over a database of 80 million images [17]. Several followup approaches have improved upon the power of binary hashing for compression [8, 13]. Most previous evaluations of binary hashing have evaluated retrieval of images similar to the query image from a large database, as opposed to multi-view image retrieval of a collection of objects.

In this work we evaluate the appropriateness of binary hashing on local descriptors for real-time multi-view image matching. Kulis and Grauman [8] tested binary hashing for matching SIFT descriptors in the Photo Tourism dataset [14]. Chandrasekhar et al. also considered using spectral hashing and other binary compression methods to match SIFT descriptors, and found the CHoG descriptor to be superior to these techniques in terms of keypoint matching error rate [4]. They do not give explicit matching times. Our evaluations differs from these in that we consider image retrieval performance as well as keypoint recognition, and consider the trade-offs between database size, matching speed and retrieval performance. We also target the case of online image retrieval for real-time visual tracking, as opposed to high-latency image queries over the network. This motivates our search for very low bit-rate descriptors, so that the database can be stored in the restricted memory space of a mobile phone.

Descriptor	Size (bits)	Matching
SIFT [9]	1024	kd-tree, L2-dist
HIPs [16]	352	Binary comparisons
PhonySIFT [18]	288	spill tree, L2-dist
BRIEF [3]	256	Hamming distance
CHoG [5]	33-90	Kullback-Leibler
Our method	16-64	Hamming distance

Figure 1. Comparison of keypoint descriptors.

3. Descriptor Compression

Most feature descriptors are matched using Euclidean distance. This requires storing the entire descriptor, and matching them using a partitioning structure for speed. Wagner *et al*. showed an improvement over the traditionally-used kd-trees by using a split forest [18]. However, they still found memory to be the limiting factor for keypoint recognition on mobile phones.

The spectral hashing compression method improves upon the speed memory requirements for matching descriptors in Euclidean space by transforming them into a space of bit-vectors [20]. Spectral hashing is a variant of locality-sensitive hashing (LSH) which attempts to embed vectors into a bit-vector space such that neighbors in Euclidean space have low Hamming distance in the binary space. Spectral hashing achieves much higher precision than LSH with the same number of bits.

Spectral hashing has three simple steps: 1) apply principal component analysis (PCA) to align all descriptors from an image to the major axes to their principal components; 2) model the set of PCA-aligned descriptors with a multivariate uniform distribution; and 3) find the K smallest eigenfunctions of the distribution. See Weiss, Torralba and Fergus's paper for more details [20]. To compress a given vector, we threshold the K eigenfunctions at zero to produce a K-bit descriptor. The spectral hash can be matched using the Hamming distance.

We show in Section 6 that even 16 bits is sufficient to effectively match keypoints using spectral hash in some cases. Figure 1 compares the size and matching method of our method with several common keypoint descriptors.

4. Descriptor Indexing

Computing the Hamming distance between two bit-vectors is quite fast. For small database sizes, linear search can be used to match descriptors. However, the matching time can be improved using an indexing structure.

We note that we can collect bit-vectors into a $K+1$-level binary tree, where the leaves correspond to K-bit vectors. To build the tree, we insert bit-vectors by starting from the root, and moving left at a zero bit and right at a one bit. We store the keypoint information in the leaf corresponding to its bit vector. Insertion is therefore linear in the number of bits. We can find all vectors in the database with a Hamming distance of at most D by traversing the tree from the root, and pruning any sub-trees whose leaves would exceed the distance threshold. Thus we have to perform less comparisons than linear search of the list of all features, since we can prune large sections of the tree without traversing to their leaves. Because the structure of the tree is fixed, there is no concern of the tree becoming unbalanced after insertion, as in the case of the kd-tree.

5. Image Retrieval

The compressed descriptor matching algorithm described in Sections 3 and 4 is useful for pose estimation with a single image. However, typically we would like to recognize several tracking targets. In this section we describe how spectral hashing can be used for multi-view image retrieval.

To match a query image to a collection of images, we propose the following simple method. Features from the query image are hashed and matched to features in the database as described in Section 4. We then rank database images by their number of matches. For greater robustness, the images with the highest number of matches can be subjected to geometric verification [6].

For some applications, it is useful to insert new images into the database for retrieval at runtime. However, we would like to avoid having to completely rebuild the database, because this can be time-consuming as the number of images increases. We have developed and tested methods, described below, to incrementally update the spectral hashing parameters to lessen the cost of inserting a new image.

When inserting an image, we need to update the PCA coefficients and the uniform distribution to reflect the expanded set of features, as described in Section 3. PCA involves finding the mean and covariance of all features from all images. However, computing the covariance is especially slow because of the number of multiplications required. To avoid re-computing the covariance of all features, we propose summing the mean and covariance from each image, computed separately. When adding a new image for retrieval, we only need to compute the mean and covariance of the features in the new image and add them to the previously stored sums. We are in effect describing the set of images with a multi-variate Gaussian mixture, with equal probability of sampling from each Gaussian.

We update the uniform distribution by finding the new bounds of the aligned descriptor space. We approximate these bounds to avoid iterating through the entire set of descriptors. To do this, we first transfer the previous bounds to the new space, by multiplying them by the inverse of the old principal components, and then by the new principal components. This approximates the bounds of the previously added vectors using the updated principal components. We then insert the new descriptors, update the bounds, and find the K minimum eigenfunctions as before.

Once the compression parameters have been updated, all descriptors must be re-hashed with the updated binary encoding. However, the actual encoding time per descriptor is quite fast, as evidenced by our evaluation (see Section 6.2).

6. Evaluation

We performed several tests using the SURF-128 descriptor [2]. This descriptor is faster to compute than SIFT, but exhibits similar performance. We chose it as a reasonably fast descriptor to compute that exhibits state-of-the-art recognition performance and uses SSD for matching. Our goal in evaluation is to show that by embedding Euclidean-space descriptors in binary vector space, we can achieve multi-view image recognition performance comparable to SSD with less space and computation time. Because the robustness of the SURF descriptor with SSD matching is well known, we focused on comparing SSD matching with hashing.

We used two multi-view datasets for our tests. The first is a set of video sequences with continuous motion used for evaluation of detectors and descriptors for visual tracking, from the University of California, Santa Barbara [7]. This dataset includes videos of six different planar tracking targets and several different motion patterns. A ground truth homography between each image and a reference pose is provided. We used the *Paris* target which has a reasonable amount of unique texture, and the first thirty frames of the "perspective distortion" sequence, which shows out-of-plane rotation of the target. To restrict our evaluation to matching of tracking targets, we whited out the regions of the images outside the tracking target. Figure 2 shows a subset of the images from this dataset.

The second dataset is a multi-view image retrieval database from the University of Kentucky with four images per object [10]. We used the first 100 images from the *UK* database for our retrieval test. The objects vary in difficulty for matching due to the varying appearance and capture conditions. Figure 4 shows sample images from this dataset.

6.1. Feature Matching Performance

To test feature matching on the *Paris* dataset, we extracted the 150 strongest SURF keypoints from each of

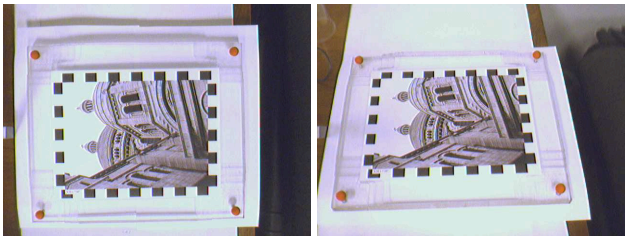

Figure 2. Two images from the *Paris* dataset, used for testing multi-view feature matching performance. The area outside of the tracking target was whited out for the evaluation.

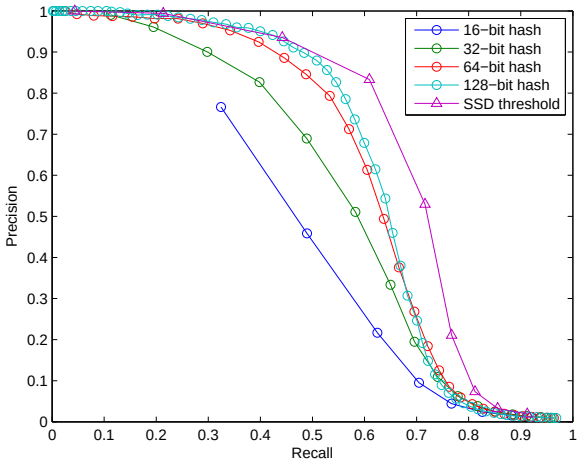

Figure 3. Precision-recall graph for feature matches on the *Paris* dataset.

the thirty images. We used the first image as the training set, and ran the remaining twenty-nine images as queries. Matches were validated by using the ground truth homographies. Features from the query image are projected back to the training image, and any matches with a re-projection error of less than five pixels are accepted.

For SSD matching, we built a kd-tree from the features in the training image, and used a range query for each query descriptor. All descriptors within the SSD threshold were treated as matches. Since SURF descriptors are normalized to have unit magnitude. we tested SSD thresholds from 0.1 to 0.9.

To test spectral hashing, we generated the hashing parameters from the descriptors in the first image, and hashed the descriptors from all images. We tried a range of bit-vector lengths from 16 to 128 bits. To match a query descriptor, we traversed the Hamming tree and accepted all training vectors within the chosen threshold. We used a range of Hamming distance thresholds, from zero to half the number of bits in the vector.

Figure 3 shows the precision-recall graph for all matching methods and parameter settings. With the highest number of bits tested, hashing achieves performance very close to that of SSD. Performance decreases gradually with the size of the bit-vector. However, the 16-bit descriptor can still achieve greater than 70% precision while returning about a third of all possible correct matches.

6.2. Image Retrieval Performance

We used images from the *UK* dataset to test our image retrieval procedure. To test the scalability of our method, we tested the number of correct image matches as the number of training images increases, from twenty images (five classes) to one hundred images (twenty-five classes). Each class has four images, and we queried all images against the database. The four images with the most feature matches were used as the query result.

SSD matching was implemented as before. We built a kd-tree on all features in the training set, and used a range query for feature matching. From the test described in Section 6.1, we determined a threshold of 0.4 to give the best performance, and thus used this threshold in our image retrieval test.

For spectral hashing, we determined the hashing parameters by incrementally adding images to the training set as described in Section 5. We kept three vector length and Hamming threshold combinations evaluated in our first test: 16-bit vectors with a threshold of zero, 32-bit vectors with a threshold of four, and 64-bit vectors with a threshold of eight. The performance with 128 bits was very similar to that of 64 bits, so we omitted that configuration from our test.

In Figure 5 we plot the percentage of correct matches as a function of the number of images in the database. Performance is generally flat for all of the methods, and the worst performance (with 16 bits) was still above 70% precision. Also, the 64-bit vector slightly outperforms SSD matching with up to eighty images in the database, and the 32-bit vector outperforms both with up to forty training images.

We also measured query time per feature as the database size increases, plotted in Figure 6. With about 150 features per image, we tested database sizes of up to 15,000 features. Query time increases with the bit-vector length. Also, the Hamming distance threshold affects query speed as the database size increases, as the 64-bit vector with a Hamming threshold of eight shows increasing query time with database size. SSD matching is most significantly affected by database size: the query time is slightly faster than 64-bit hash with up to sixty training images, and then sharply increases. With one hundred database images, the 16-bit hash had an average per-feature query time of 2 μs, and the 32-bit hash had an average per-feature query time of 170 μs. In comparison, we have a smaller database size than in the work of Taylor *et al*. [15] and faster matching time than in the work of Wagner *et al*. [19].

Figure 4. Images from the *UK* dataset, used for testing multi-view image retrieval performance.

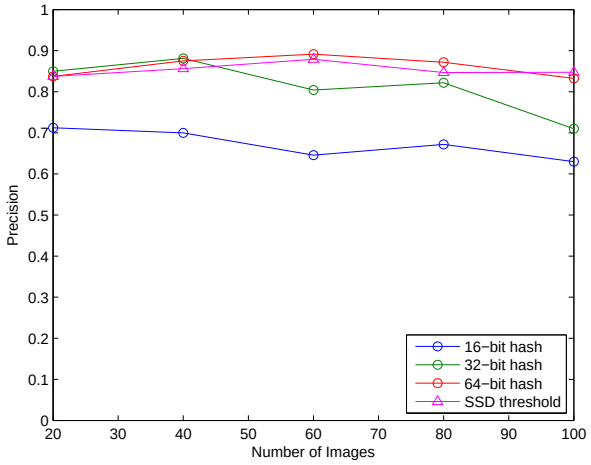

Figure 5. Image retrieval precision using the *UK* dataset.

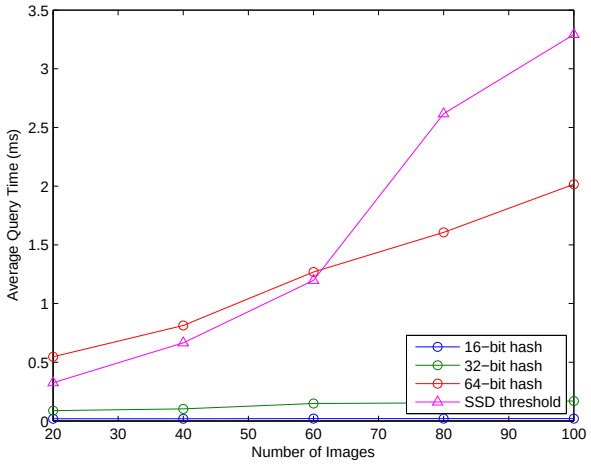

Figure 6. Average query time per keypoint using the *UK* dataset.

7. Conclusions and Future Work

In this paper we show how spectral hashing can be used to compress invariant descriptors for multi-view matching down to a very small bit-vector, with little loss in matching performance. We developed an indexing structure for speeding up range queries using the Hamming distance. We also developed a way to adapt spectral hashing to support insertion of new training images to the compression structure.

Our evaluation of both feature matching and image retrieval performance demonstrates that a good level of performance can be achieved with as little as 16 bits per descriptor. We determined the 32-bit descriptor to have the best compromise between matching performance, descriptor size, and matching speed. We show real-time performance with one hundred images in the database, and thousands of features. In comparison to recent state-of-the-art work on real-time image retrieval that matches hundreds of images [12], we achieve recognition on the same order of images with a much simpler approach, due to a faster matching procedure and smaller descriptors.

Our overall feature matching approach is not yet suitable for implementation on a mobile phone for real-time tracking, since the SURF detector and descriptor are too slow to compute. In the future we would like to test spectral hashing for matching of a reduced descriptor such as PhonySIFT [18], which has been demonstrated to have real-time performance on a mobile platform. We believe such a combination would be perfect for a robust and scalable mobile SLAM system. With a fast and scalable descriptor index, we could build and store a very large database of features and images on the phone, without having to rely on external storage of the feature database and queries over the network as in previous work [1, 5].

Acknowledgements

This work was partially supported by NSF CAREER grant IIS-0747520.

References

[1] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *ISMAR '09: Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 73–82, Washington, DC, USA, 2009. IEEE Computer Society. 5

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008. 1, 3

[3] M. Calonder, V. Lepetit, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision*, 2010. 2

[4] V. Chandrasekhar, M. Makar, G. Takacs, D. Chen, S. S. Tsai, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, , and B. Girod. Survey of sift compression schemes. In *International Workshop on Mobile Multimedia Processing (WMMP)*, August 2010. 2

[5] V. Chandrasekhar, Y. Reznik, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. Quantization schemes for low bitrate compressed histogram of gradients descriptors. In *IEEE International Workshop on Mobile Vision (in conjunction with CVPR 2010)*, 2010. 2, 5

[6] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007. 3

[7] S. Gauglitz, T. Höllerer, and M. Turk. Dataset and evaluation of interest point detectors for visual tracking. Technical Report 2010-06, Department of Computer Science, University of California, Santa Barbara, 2010. 3

[8] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *12th. International Conference on Computer Vision (ICCV)*, 2009. 2

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004. 1, 2

[10] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. 3

[11] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2007. 1, 2

[12] J. Pilet and H. Saito. Virtually augmenting hundreds of real pictures: An approach based on learning, retrieval, and tracking. Technical report, IEEE Virtual Reality, 2010. 5

[13] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1509–1517. 2009. 2

[14] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846, New York, NY, USA, 2006. ACM. 2

[15] S. Taylor and T. Drummond. Multiple target localisation at over 100 fps. In *British Machine Vision Conference*, September 2009. 4

[16] S. Taylor, E. Rosten, and T. Drummond. Robust feature matching in 2.3μs. In *IEEE CVPR Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond*, June 2009. 1, 2

[17] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. pages 1 –8, jun. 2008. 2

[18] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR '08: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134, Washington, DC, USA, 2008. IEEE Computer Society. 2, 5

[19] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *ISMAR '09: Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 57–64, Washington, DC, USA, 2009. IEEE Computer Society. 2, 4

[20] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1753–1760. 2009. 1, 2