Binary Coherent Edge Descriptors

C. Lawrence Zitnick

Microsoft Research, Redmond, WA

Abstract. Patch descriptors are used for a variety of tasks ranging from finding corresponding points across images, to describing object category parts. In this paper, we propose an image patch descriptor based on edge position, orientation and local linear length. Unlike previous works using histograms of gradients, our descriptor does not encode relative gradient magnitudes. Our approach locally normalizes the patch gradients to remove relative gradient information, followed by orientation dependent binning. Finally, the edge histogram is binarized to encode edge locations, orientations and lengths. Two additional extensions are proposed for fast PCA dimensionality reduction, and a min-hash approach for fast patch retrieval. Our algorithm produces state-of-the-art results on previously published object instance patch data sets, as well as a new patch data set modeling intra-category appearance variations.

1 Introduction

The ability to describe an image patch is critical to many recognition algorithms. Image patches can be used to find correspondences between varying viewpoints of an object [1,2,3,4], or to represent parts of object categories [5,6,7]. Typically, a desirable patch descriptor is robust to illumination changes, moderate pose variation, and intra-category appearance variation.

A standard approach to describe a patch is the use of Histograms of Gradients (HoG), [1,7,8,9,10,11]. A HoG is defined as the histogram of image gradients over a combination of positions, orientations and scales. Examples include the SIFT [1] and GLOH [9] interest point descriptors, which have been shown to be very effective for object instance recognition. Similar approaches have been applied to describe object category parts [7,12]. After creating histograms from local pixel gradients, standard HoG approaches rely on a global normalization step to account for variations in illumination. However, these descriptors are still sensitive to the relative magnitudes of gradients. In many scenarios such as intra-category appearance variation and partial illumination changes the relative gradient magnitudes do vary, resulting in reduced matching performance. Several approaches [1,11,12] use truncated normalization to help reduce this sensitivity.

In this paper, we propose an image patch descriptor based on the location, orientation, and length of edges, and not their relative gradient magnitudes. We hypothesize that the presence and not magnitude of edges provides an informative measure of patch similarity that is robust not only to illumination and pose changes, but intra-category appearance variation. Our descriptor encodes the presence or absence of edges using a binary value for a range of possible edge positions and orientations. In addition the locally linear length of an edge is used to differentiate sets of coherent edges aligned perpendicular to the edge orientation from shorter edges resulting from textures. Our approach consists of three main steps: First, the image patch gradients are locally normalized to remove variations in relative gradient magnitudes. Second, the normalized gradients are binned using the position, orientation and local linear length of an edge. Finally, the normalized gradient histogram is binarized to encode the presence of edges.

In addition to the basic approach we propose two extensions: a fast method for dimensionally reduction using binary vectors and PCA, and a min-hash feature representation for efficient retrieval. The approach is tested using a previously published [11] ground truth object instance data set to test its invariance to illumination and pose changes. A new data set is provided to test invariance to intra-category appearance variation. In both cases, state-of-the-art results are achieved, with significant increases in accuracy over traditional approaches such as SIFT [1], GLOH [9] and variants of Daisy descriptors [10,11].

The rest of the paper is organized as follows: In the next section we describe previous work, followed by our basic approach. In Section 4 we discuss extensions to our algorithm. Finally results are provided in Section 5 following by a conclusion and discussion.

2 Previous Work

There exists a large body of previous work on image patch descriptors [13]. The SIFT [1] descriptor popularized the HoG approach and introduced several optimizations, including truncated normalization and ratio tests. Several follow up papers have improved on the SIFT desciptor using PCA [14], radial binning [9] and "daisy" binning [11,15]. Spatial binning parameters have also been learned from training data [10,11]. Geometric Blurring [8] proposed blurring the gradients using a spatially varying blur kernel based on the distance to the center of the patch. SURF [16] uses Harr wavelets instead of gradients to describe image patches. Another approach is to use generative models to learn the statistics of image patches [17].

Image patches have also been described and classified using randomized trees [18,19] and boosting [20] to aid in detecting object classes.

Gradients are commonly used for category part representation. Felzenszwalb et al. [7] and Dalal and Triggs [12] use HoGs for object category detection, while others such as Crandall et al. [6] use binary edge detection. PCA on image intensities has also shown good results in Fergus et al. [5].

3 Binary Edge Descriptor

Our descriptor relies on the detection of edges in an image patch. It is assumed that the presence of edges remains consistent across matching image patches, even if their relative magnitudes do not. Thus, we describe an edge based on its orientation, position and length, and not its gradient magnitude. For instance see Figure 1(a). Both patches share similar edge structure, but the relative gradient magnitudes vary significantly.

Our method is split into three stages: gradient normalization, edge aggregation and binarization. Gradient normalization removes differences in relative gradient magnitudes between edges. It is worth noting that we locally normalize gradients to remove relative differences in magnitude, instead of a global normalization [1] that only accounts for global gain and offset differences. Next, the gradients are aggregated into bins, after which a binarization process labels the bins with highest contribution. Before we describe these three stages, we define our initial gradient orientations and magnitudes.

The descriptor is computed from a $n \times n$ square patch of pixels. The intensity of a pixel p at location (x_p, y_p) is denoted $f(p)$ or $f(x_p, y_p)$. The horizontal gradient $f_x(p)$ of the pixel is equal to $f(x_p+1, y_p) - f(x_p, y_p)$ and similarly for the vertical gradient $f_y(p)$. The magnitude of the gradient for pixel p is the Euclidean norm of its gradients, $g_p = \|[f_x(p)\ f_y(p)]^{\mathrm{T}}\|_2$. The orientation is defined as $\theta_p = \arctan(f_y(p)/f_x(p))$. To help remove noise and sampling artifacts a small amount of Gaussian blur ($\sigma = 0.5$) is applied to the patch before computing the gradients and orientations.

3.1 Gradient Magnitude Normalization

Our goal for gradient normalization is to maintain the gradient profiles while removing the relative height differences between the gradient peaks. An efficient method to solve this problem is to normalize the gradients based on the average gradient magnitude in a local spatial neighborhood. We compute the average Gaussian weighted gradient magnitude $\bar{g}_p$ in a spatial neighborhood N of p using

$$\bar{g}_p = \sum_{q \in N} g_q \mathcal{N}(q; p, \sigma_s), \tag{1}$$

where $\mathcal{N}$ is the standard normal distribution. The normalized gradients $\hat{g}_p$ are computed using the ratio of the original gradients and the average gradients,

$$\hat{g}_p = \frac{g_p}{\max(\bar{g}_p, \epsilon)}, \tag{2}$$

where $\epsilon = 4$ is used to ensure the magnitude of $\bar{g}_p$ is above the level of noise. In our experiments the spatial standard deviation is set to $\sigma_s = 3$. Examples of the normalized gradients are shown in Figure 1(a). We also experimented with including orientation to compute the average gradients in three dimensions. This avoids edges with large gradient magnitudes inhibiting the gradients of nearby edges with different orientations. However, this computationally more expensive approach did not improve the accuracy of the final descriptor.

3.2 Edge Aggregation

The next stage of our approach aggregates the normalized gradients into bins defined by an edge's position, orientation and local linear length. We align the

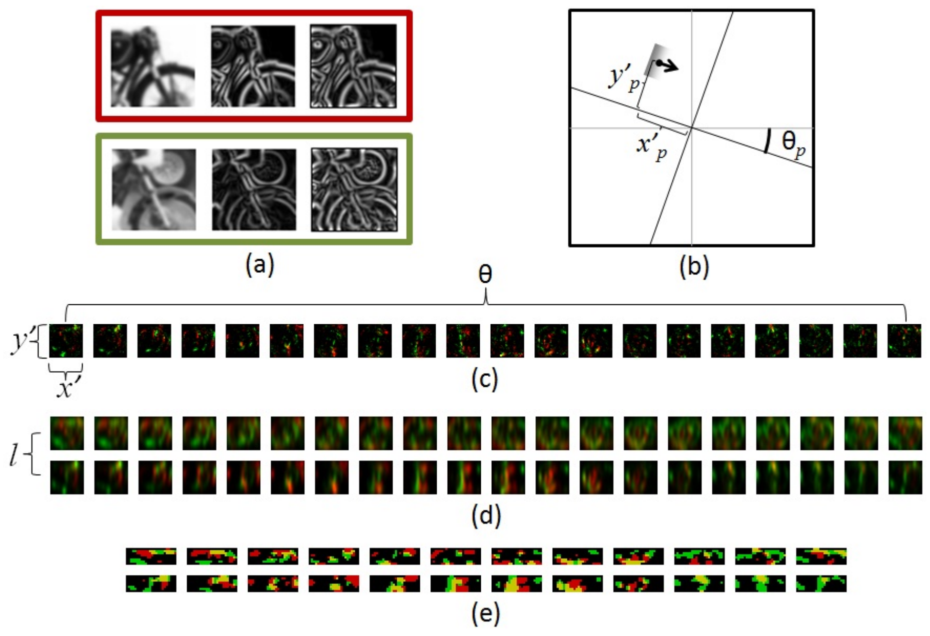

Fig. 1. Processing pipeline: (a) Two matching patches (left to right): Original patch, gradients g_p and normalized gradients $\hat{g}_p$, (b) an illustration of the coordinate frame used for the orientation dependent binning, (c) the edge histogram with the top patch shown in red and bottom patch shown in green (yellow denotes agreement), (d) the histogram after blurring and spliting the edges into two sets of bins based on the edge's local linear length, (e) final binarized descriptor, red is the top patch, green is the bottom patch and yellow denotes agreement

spatial binning with the gradient's orientation to allow for the descriptor's robustness to vary perpendicular and parallel to an edge. Orientation dependent sampling also aids in the detection of coherent edges (as shown later), i.e. sets of similarly orientated gradients aligned perpendicular to the gradient orientation. This varies from previous approaches [1,9,11] that define the spatial binning independent of the orientation. Specifically as illustrated in Figure 1(b), we define a new coordinate frame (x'_p, y'_p) for each pixel p at position (x_p, y_p) depending on its orientation θ_p equal to

$$\begin{bmatrix} x'_p \\ y'_p \end{bmatrix} = \mathbf{R}(\theta_p) \begin{bmatrix} x_p \\ y_p \end{bmatrix}, \tag{3}$$

where $\mathbf{R}(\theta_p)$ is a standard 2D rotation matrix. We assume the origin $(0, 0)$ is at the center of the patch. Using (x'_p, y'_p, θ_p) we define our binning on a $b_{x'} \times b_{y'} \times b_\theta$ resolution grid creating a histogram $H(x', y', \theta)$. In practice we use $b_{x'} = 32$, $b_{y'} = 32$ and $b_\theta = 20$. When assigning the values $\hat{g}_p$ to each bin according to (x'_p, y'_p, θ_p), we use the standard linear soft binning approach using bilinear interpolation [1]. An example of the resulting bin values can be seen in Figure 1(c).

Detecting coherent edges: Above, we aggregated the normalized gradients into a fixed number of bins in a 3D Histogram. Specifically, we split the vertical y' dimension into $b_{y'}$ bins, capturing edges $1/b_{y'}$ the length of the patch. Many edges run the entire length of the patch. The discriminability of the descriptor could be increased if long coherent edges could be distinguished from shorter texture edges. A simple approach to estimate edge length $L(x', \theta)$ for an edge at position x' and orientation θ is to sum the vertical bins perpendicular to its gradient's direction,

$$L(x', \theta) = \sum_{y'} H(x', y', \theta). \tag{4}$$

If we assign a value of $l_p = L(x', \theta)$ to every gradient $\hat{g}_p$ we may create a four dimensional histogram $H(x', y', \theta, l)$. In our experiments we found discretizing the edge lengths into two bins, $b_l = 2$, results in an effective separation of coherent edge gradients and short texture edges, as shown in Figure 1(d). Specifically, we compute a delta function $\Delta(l_p)$ equal to

$$\Delta(l_p) = \max(0, \min(1, \frac{l_p - \alpha}{\beta})). \tag{5}$$

where the values α and β where set to 2 and 8 respectively. Other sigmoid functions may also be used, but this linear form provides efficient computation. The normalized gradient values $\hat{g}_p$ are split between the two edge length bins using $\Delta(l_p)$ and $1 - \Delta(l_p)$ as weights.

3.3 Binary Representation

Given a 4D histogram $H(x', y', \theta, l)$ we want to determine the edges present in the patch, while providing robustness to small changes in position and orientation. Robustness is provided by applying a small amount of blur to the histogram. We apply Gaussian blurring in the x', y' and θ dimensions with standard deviations of $\sigma_{x'}$, $\sigma_{y'}$ and σ_θ respectively. Optimizing over possible values of $\sigma_{x'}$, $\sigma_{y'}$ and σ_θ we empirically found values of $\sigma_{x'} = 1$, $\sigma_{x'} = 3$ and $\sigma_\theta = 1$ to work well. An increased amount of blur is applied to the $\sigma_{y'}$ dimension parallel to the edges, since this dimension proved less informative in our experiments, see Section 5. An example of the blurred histogram is shown in Figure 1(d).

Before binarizing edges in the histogram, we first reduce its resolution to $n_{x'} \times n_{y'} \times n_\theta \times n_l$ using sub-sampling. Empirically we found dimensions of $n_{x'} = 24$, $n_{y'} = 8$, $n_\theta = 12$, and $n_l = 2$ for the x', y', θ and l dimensions respectively to provide good results. Experiments for various values of $n_{x'}$, $n_{y'}$, n_θ, n_l and are shown in Section 5.

We binarize the sub-sampled histogram's values by assigning a value of 1 to the top τ percent of the bins with highest values, and 0 to the others. To reduce bias in the detection of longer edges over texture edges, we perform binarization independently for both sets of edge length bins. The final binarized descriptor is denoted D, and an example is shown in Figure 1(e). The binarization process provides nearly full invariance to edge magnitudes. It also provides computational advantages when reducing the descriptor's dimensionality as we discuss in Section 4.1. In our experiments $\tau = 20\%$. Results using other values are shown in Section 5. In practice, several efficient $O(n)$ methods for finding the top τ percent may be used and are commonly referred to as "selection algorithms" [21].

4 Extensions

In this section, we describe two separate extensions to our basic approach for reducing the dimensionality of our descriptor using PCA and min-hash.

4.1 Dimensionally Reduction Using PCA

The size of our descriptor D is $n_{x'} \times n_{y'} \times n_\theta \times n_l$, which for the values described above is 4,608 dimensions. This is far larger than standard descriptors such as SIFT using 128 dimensions. The difference isn't quite as dramatic if it is considered that our descriptors are binary. For instance, we could store our descriptor in the same space as 144 32-bit floating point numbers. Furthermore, comparison between descriptors can be done efficiently using bit-wise *xor* functions [22,23,24].

In this section we explore dimensionally reduction using Principal Component Analysis (PCA). It has been shown [9,11,14] that using PCA can both decrease the dimensionally of a descriptor and improve accuracy. We perform PCA using a standard approach to compute K basis vectors. The training dataset Yosemite provided by [11] was used to learn the basis functions. Using real-valued descriptors, the cost of projecting an M dimensional descriptor using K basis functions uses MK multiplications and additions, which can be computationally expensive for large descriptors.

To increase efficiency, we can take advantage of two properties of our descriptors; they are binary and neighboring values typically have the same values, Figure 1(e). As a result, we can use a technique similar to integral images to efficiently project our descriptors by pre-computing the following values

$$\mathrm{w}^{\Sigma}_{k,i} = \sum_{j<i} \mathrm{w}_{k,j}, \tag{6}$$

where $\mathrm{w}_{k,i}$ is the ith value in the kth basis vector. Thus, $\mathrm{w}^{\Sigma}_{k,i}$ is the sum of all values in w_k before the ith entry. To compute the reduced dimensional descriptor D^* the kth projection of D is computed as

$$D^*_k = \sum_i (D_{i-1} - D_i)\mathrm{w}^{\Sigma}_{k,i}. \tag{7}$$

Since $(D_{i-1} - D_i)$ is only nonzero when neighboring values aren't equal, the total amount of computation is greatly reduced. In our experiments, on average only 10% of neighboring values were not equal when parsing the descriptor using an x', y', θ and l ordering of the dimensions, resulting in just $0.1{*}MK$ adds on average to project onto the PCA vectors. To handle boundary conditions, an additional entry has to be added to the end of all descriptors with a value of 0. Results using PCA dimensionality reduction can be found in Section 5.

4.2 Min-hash for Fast Patch Retrieval

We propose using min-hash as an efficient means for finding similar descriptors. Previous works use min-hash [25] for image retrieval and clustering [26,27]. Locality sensitive hashing, semantic hashing and binary coding [28,23,24] have also been used for image retrieval. Hashing techniques used in conjunction with inverse look-up tables provide a fast and scalable method for finding similar points in high dimensional spaces with certain probabilistic guarantees. In particular, the min-hash technique has the property that the probability of two hashes being identical is equal to the Jaccard similarity. The Jaccard similarity is the cardinality of the intersection of two sets divided by their union's cardinality. In our task, the elements of the set are the indices assigned to 1 by our descriptor. A min-hash is found by creating a random permutation of the set of possible indices. The smallest permutated index with a value of one in a descriptor is its resulting hash value [25]. Multiple hashes can be generated for a single descriptor using different random permutations. Given a set of descriptors with hashes, an inverse lookup table can be created to efficiently find descriptors with equal hash values. If enough hashes are shared between two descriptors, they are said to "match". The advantage of hashing over simple quantization such as vocabulary trees [2] and kd-trees is the matching accuracy is proportional to the number of hashes stored per descriptor and not fixed based on the amount of quantization. In this regard it is similar to using randomized kd-trees [29] or multiple quantizations, except the quantized values can be efficiently computed without traversing a tree.

In order to increase the uniqueness of a hash, hashes can be concatenated into sketches. The size of the sketch refers to the number of hashes used to create it. If the Jaccard similarity between two patches f and f' is $J(f, f')$, the probability of two sketches being identical is $J(f, f')^k$, where k is the sketch size. Min-hash is increasingly effective if the Jaccard similarity between matching images is high and is low for non-matches. In Figure 4(a), we see the density functions for matching and non-matching image pairs with respect to the Jaccard similarity. Since our descriptor produces significant separation between the two distributions and it is binary, it is a good candidate for the min-hash algorithm. We present results using the min-hash approach with various sketch sizes and numbers of sketches in Section 5.

5 Experimental Results

In this section, we provide experimental results on three datasets. The Liberty and Notre Dame datasets [11] contain image patches generated from Difference of

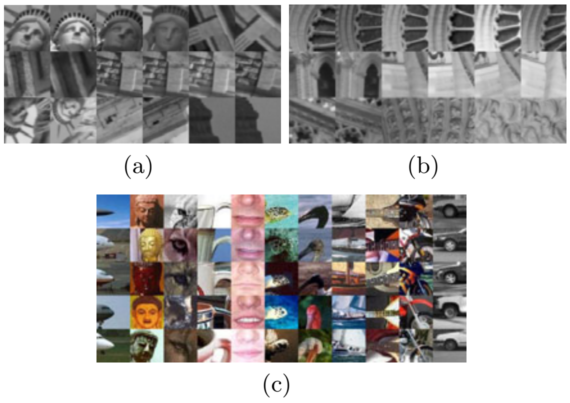

Fig. 2. Examples of matching patches from the (a) Liberty, (b) Notre Dame and (c) category datasets

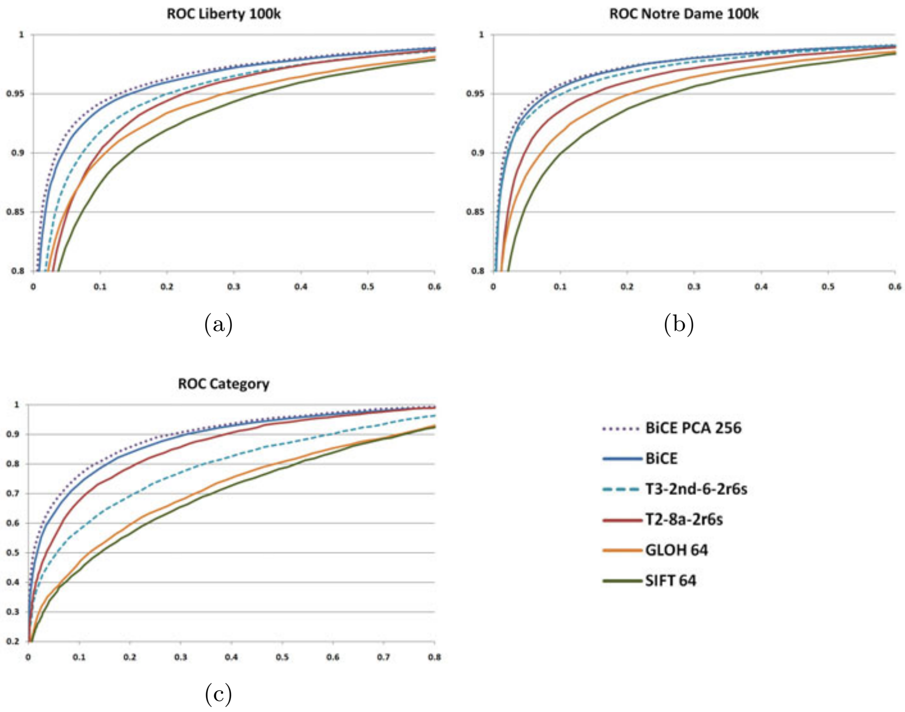

Fig. 3. ROC curves for (a) Liberty, (b) Notre Dame and (c) Category datasets. Notice the plotted ranges vary from (a,b) to the more difficult dataset of (c).

Table 1. Liberty, Notre Dame and Category dataset accuracies for SIFT [1], Gloh [9], T2-8a-2r6s, T3-2nd-6, and T3-2nd-6 [11] compared to our approach BiCE. Errors at 95 % recall, and Equal Error Rates (EER) are given. (B) indicates binary dimensions.

		Liberty		Notre Dame		Category	
Method	Dimensions	95% Error	EER	95% Error	EER	95% Error	EER
SIFT [11]	128	35.09	-	26.10	-	-	-
SIFT 64	128	33.38	11.51	26.31	10.03	87.37	32.53
GLOH 64	272	28.38	10.27	20.46	8.87	86.69	31.25
T2-8a-2r6s	104	22.37	9.89	14.70	7.57	54.49	20.59
T3-2nd-4 [11]	416	19.36	-	10.50	-	-	-
T3-2nd-6	624	20.08	8.89	10.15	6.35	74.34	25.70
T3-2nd-4 PCA [11]	37	17.24	-	9.71	-	-	-
T3-2nd-6 PCA [11]	42	17.14	-	9.49	-	-	-
BiCE	4608 (B)	14.47	7.50	8.34	6.01	48.66	17.77
BiCE PCA	256	12.76	7.03	7.46	5.72	45.04	16.74
BiCE PCA	128	13.85	7.24	8.01	5.95	47.69	17.44
BiCE PCA	64	15.82	7.90	9.97	6.50	49.12	18.94
BiCE PCA	32	20.15	9.09	14.37	7.58	54.51	20.84

Gaussian interest point detectors [1] from the Statue of Liberty and Notre Dame cathedral, as shown in Figure 2(a,b). Pairs of matching image patches are verified using structure from motion [4]. These datasets are effective for measuring a patch descriptor's robustness to lighting variation and changes in viewpoint.

We created an additional dataset to measure robustness to intra-category appearance variation, as shown in Figure 2(c). The category dataset consists of 20 collections of 64×64 patches extracted from the Caltech 256 [30] dataset. Each collection of patches is selected by humans from a single category centered on the same part of the object, e.g. the back wheel of a motorcycle, the head of a turtle, etc. From these sets, 12,800 positive patch pairs are split to create testing and training datasets. An equal number of negative patch pairs are also generated using random patch selection. The dataset is available from the author's website.

Table 1 shows the results of various patch descriptors on the three datasets. We compare our approach Binary Coherent Edge descriptor (BiCE), to SIFT [1], Gloh [9], and several state-of-the-art descriptors T2-8a-2r6s, T3-2nd-4, T3-2nd-6 from [11]. Error rates at 95% recall and Equal Error Rates (EER) are given. The EER is the point on the ROC curve where the percentage of false positives and false negatives are equal. Since we are using 64×64 patches we also computed SIFT 64 and Gloh 64 using their standard resolutions for spatial binning, but with the full resolution patches for a fair comparison. Results of our descriptor using smaller patches and other variations are shown in the next section. Results using PCA with various dimensions are also shown. Parameters are kept constant for all experiments using the values stated in previous sections. Running times for computing a descriptor were approximately 11ms for BiCE, 2ms for SIFT and 14ms for T3-2nd-6 on a 2.4GHz Intel PC. The code for BiCE is only partially optimized.

The best results are found across all datasets using BiCE with PCA and 256 dimensions, followed closely by BiCE without PCA. The results for T3-2nd-4 and T3-2nd-6 with PCA also perform well. However, rotating these descriptors using PCA in high dimensional spaces can be computationally expensive. It is worth noting that T2-8a-2r6s does relatively better on the category dataset than other previous methods. We hypothesize this is due to the inhibition technique used to compute orientation binning.

5.1 Parameter Exploration

In this section, we explore various adjustments and parameter changes to the previously described approach. The results are summarized in Table 2. The first set of figures shows the result of various sampling densities on the histogram H to get our final descriptor D. The results show that additional sampling in the y' dimension does not provide additional accuracy. As the sampling rate decreases the accuracies slowly decrease. Even with only 432 binary dimensions (54 bytes of storage) the accuracies still outperform previous techniques. The value of τ is varied from 10% to 30%, with only minor differences in accuracies. The removal of the edge length dimension increases the error rate by approximately 6% at 95% recall. The direct use of normalized continuous values sampled from H instead of using binarization significantly increases the 95% error rate to 27.42%. Similar to the binarization stage, the bins corresponding to different edge lengths were normalized independently. Normalizing all values together produces worse results. We also tried binarizing T3-2nd-6 [11] and SIFT [1] features using our simple approach, but improved results were not achieved. Other more sophisticated approaches to binarization could produce better results [23,24]. Finally we tested the descriptor's invariance to the initial patch size. As the patch size decreases, the accuracies are slightly better (32×32) or slightly worse (18×18).

5.2 Min-hash

The results using the min-hash approach from Section 4.2 are summarized in Table 3. ROC curves for a subset of the results can be seen in Figure 4, with BiCE providing an upper bound on the accuracies. The "% Match" and "% Non-match" columns indicate the probability of an descriptor having a corresponding hash value if it is a matching or non-matching descriptor. For instance, if a dataset had 1 million descriptors with most being non-matches, we would find on average 155,500 descriptors in each entry of the inverse lookup table using a sketch of size 1. As we can see, sketches of larger size are advantageous to minimize collisions. However, larger sketches also require more hashes to be stored to find collisions with correct matches. The right tradeoffs are application dependent. It is interesting to note that the min-hash approach produces

Table 2. Variations of parameters and methods on the BiCE baseline algorithm. This includes different descriptor sizes, differing values of τ, removal of edge length information, using continuous values instead of binary and using various patch sizes. (B) denotes binary dimensions.

		Liberty	
Method	Dimensions	95% Error rate	EER
BiCE baseline	4608 (B)	14.47	7.50
BiCE $n_{x'} = 24, n_{y'} = 24, n_\theta = 12, \sigma_{y'} = 1$	13824 (B)	14.68	7.67
BiCE $n_{x'} = 16, n_{y'} = 4, n_\theta = 8$	1024 (B)	15.22	7.72
BiCE $n_{x'} = 12, n_{y'} = 3, n_\theta = 6, \sigma_{x'} = 1.5, \sigma_{y'} = 4, \sigma_\theta = 1.5$	432 (B)	16.27	7.90
BiCE $\tau = 10\%$	4608 (B)	16.28	7.82
BiCE $\tau = 15\%$	4608 (B)	14.86	7.52
BiCE $\tau = 30\%$	4608 (B)	14.46	7.63
BiCE $n_l = 1$	2304 (B)	20.36	9.55
BiCE Continuous	4608	27.42	10.27
T3-2nd-6 Binary, $\tau = 20\%$	624 (B)	20.00	8.89
SIFT Binary, $\tau = 20\%$	128 (B)	39.65	13.88
BiCE 32×32 patch	4608 (B)	13.86	7.41
BiCE 18×18 patch	4608 (B)	16.03	7.84

Table 3. Error rates at 95% recall and Equal Error Rates (EER) for various sketch sizes and numbers of sketches on the Liberty dataset. The percentage of match and non-match image patches sharing a sketch on average.

		Liberty			
Sketch size	Number of sketches	95% Error rate	EER	% Match	% Non-match
1	32	42.03	19.26	40.47	15.55
1	64	33.96	11.13		
2	64	60.95	17.08	18.57	2.91
2	128	27.78	11.79		
3	128	44.36	19.67	9.23	0.65
3	256	47.94	12.14		
4	256	50.60	17.62	4.75	0.15
4	512	37.53	15.37		

similar accuracies to SIFT using 128 sketches of size 2 or 64 sketches of size 1. Hashing techniques are ideal for applications that can handle some degradation in matching accuracy for gains in efficiency, such as large scale image clustering and near-duplicate image search [26,27].

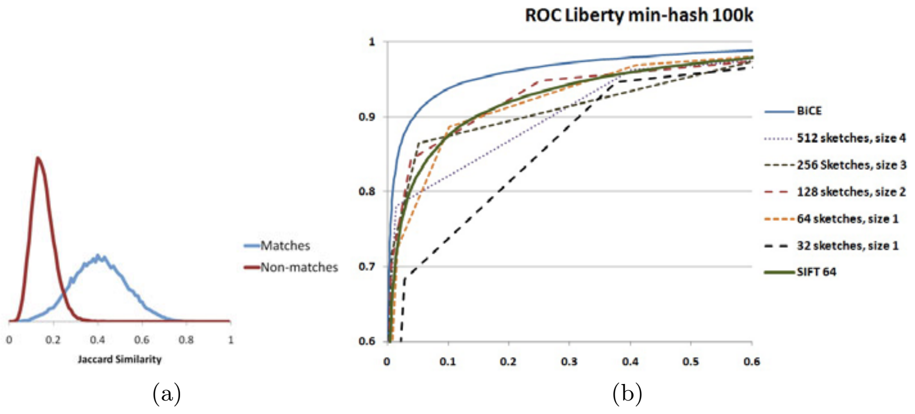

Fig. 4. (a) Density functions for matching and non-matching image pairs with respect to their Jaccard similarity, (b) ROC curves for various numbers of sketches and sizes. BiCE provides an upper bound on the accuracy of the min-hashing approaches.

6 Discussion and Conclusion

In this paper, we have developed a simple and effective image patch descriptor that provides state-of-the-art results. The descriptor encodes edge position, orientation, and local linear length, but not relative gradient magnitudes. We describe two techniques for dimensionality reduction using PCA and min-hash. Min-hash also provides a method for efficient patch retrieval.

In designing the descriptor, we experimented with other edge information such as curvature and distinguishing between even and odd edges. However, these approaches did not yield improved results. For category recognition, it can be important to be invariant to edge polarity, which our descriptor is not. It is still an open question on how to encode robustness in situations where it is useful while not providing full invariance when polarity is informative.

Finally, our edge descriptor might be invariant to relative gradient magnitudes, but interest point detectors are generally not with some exceptions [31]. An area of future work is to develop a corresponding interest point detector for sparse sampling that is robust to relative gradient magnitude and intensity changes.

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60, 91–110 (2004)
2. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR, pp. 2161–2168 (2006)
3. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. IJCV 66 (2006)
4. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. ACM Transactions on Graphics 25, 835–846 (2006)
5. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: CVPR (2003)

6. Crandall, D., Huttenlocher, D.: Weakly supervised learning of part-based spatial models for visual object recognition. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 16–29. Springer, Heidelberg (2006)
7. Felzenszwalb, P., Mcallester, D., Ramanan, D.: A discriminatively trained, multi-scale, deformable part model. In: CVPR (2008)
8. Berg, A.C., Malik, J.: Geometric blur for template matching. In: CVPR, pp. 607–614 (2001)
9. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE PAMI 27, 1615–1630 (2005)
10. Winder, S.A.J., Brown, M.: Learning local image descriptors. In: CVPR (2007)
11. Winder, S., Hua, G., Brown, M.: Picking the best daisy. In: CVPR, pp. 178–185 (2009)
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, pp. 886–893 (2005)
13. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE PAMI 27, 1615–1630 (2005)
14. Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: CVPR, pp. 506–513 (2004)
15. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR (2008)
16. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
17. Osindero, S., Hinton, G.E.: Modeling image patches with a directed hierarchy of markov random fields. In: NIPS 20 (2008)
18. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. IEEE PAMI 28, 1465–1479 (2006)
19. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: CVPR (2008)
20. Babenko, B., Dollar, P., Belongie, S.: Task specific local region matching. In: ICCV (2007)
21. Cormen, T.H.: Introduction to Algorithms. MIT Press, Cambridge (2001)
22. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large databases for object recognition. In: CVPR (2008)
23. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. of Approximate Reasoning (2009)
24. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: NIPS (2009)
25. Broder, A.Z.: On the resemblance and containment of documents. In: Compression and Complexity of Sequences (SEQUENCES 1997), pp. 21–29. IEEE Computer Society, Los Alamitos (1997)
26. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: British Machine Vision Conference (2008)
27. Chum, O., Perdoch, M., Matas, J.: Geometric min-hashing: Finding a (thick) needle in a haystack. In: CVPR, pp. 17–24 (2009)
28. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (2009)
29. Silpa-Anan, C., Hartley, R.: Optimised KD-trees for fast image descriptor matching. In: CVPR, pp. 1–8 (2008)
30. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
31. Mikolajczyk, K., Zisserman, A., Schmid, C.: Shape recognition with edge-based features. In: British Machine Vision Conference (2003)