Transform Coding for Fast Approximate Nearest Neighbor Search in High Dimensions

Jonathan Brandt
Adobe Systems, Inc.
jbrandt@adobe.com

Abstract

We examine the problem of large scale nearest neighbor search in high dimensional spaces and propose a new approach based on the close relationship between nearest neighbor search and that of signal representation and quantization. Our contribution is a very simple and efficient quantization technique using transform coding and product quantization. We demonstrate its effectiveness in several settings, including large-scale retrieval, nearest neighbor classification, feature matching, and similarity search based on the bag-of-words representation. Through experiments on standard data sets we show it is competitive with state-of-the-art methods, with greater speed, simplicity, and generality. The resulting compact representation can be the basis for more elaborate hierarchical search structures for sub-linear approximate search. However, we demonstrate that optimized linear search using the quantized representation is extremely fast and trivially parallelizable on modern computer architectures, with further acceleration possible by way of GPU implementation.

1. Introduction

Finding nearby points among a large set in high dimensions is at the heart of many important computer vision problems. Examples include nearest neighbor classification, image similarity search, and feature matching. The long-standing difficulty in finding an exact nearest neighbor in high dimensions has led to the use of approximate algorithms, as well as various domain-specific approaches.

Recently, image and video retrieval has been the subject of intense study with numerous practical applications. For retrieval, the number of points to search is usually much larger than can be held in system memory, which has led to the development of compressed representations, typically designed specifically for a given task.

In this paper, we propose a very general quantization-based framework for high dimensional nearest neighbor search, founded on well-established results in signal representation and compression, and demonstrate its effectiveness in a diverse range of computer vision tasks. The approach is inherently lossy, and is therefore an approximate approach to nearest neighbor. However, the representation degrades gracefully with increased compression, so it is straightforward to select the appropriate tradeoff between compression and accuracy for the task at hand.

2. Related Work

Despite prolonged study, the problem of efficiently finding nearby points in high dimensions remains open [4]. This difficulty has led to the development of approximate solutions, such as ANN [1]. However, beyond a few dozen dimensions, such methods tend to break down.

In the past decade, locality-sensitive hashing (LSH) has emerged as an alternative to ANN that has been shown to be effective in higher dimensions. (See [13] for a survey of LSH.) However, due to its randomized nature, LSH does not generally produce compact codes. Code compactness is important in the context of large-scale retrieval as memory size is often the primary determinant of system performance. To address this problem, Torralba et al [14] proposed learning compact codes, both by boosting, and through restricted Boltzman machines, and were able to demonstrate significant improvement over LSH.

Spectral hashing [16] improved on [14] by posing the binary hash construction problem as one of graph labeling and employed spectral relaxation as a tractable approximation to an otherwise NP-hard problem. Spectral hashing achieves code efficiency comparable to the aforementioned learning-based approaches with much greater simplicity, and is a significant improvement over LSH when applied to the GIST descriptor under the Euclidean norm.

Although hashing techniques are in general useful for near-duplicate search, they are less effective at ranged searches, where it is necessary to explore a potentially large neighborhood of a point, and distance estimation in the original space is typically not possible using the hash representation.

Following earlier work [5, 6], Jégou et al [7] approach the Euclidean nearest neighbor problem as one of efficient quantization. They argue that in the context of retrieval, what is needed is to accurately estimate inter-point distances while representing the points with a limited number of bits. That is, we not only want a compact representation, but we additionally want to be able to estimate distances using the compact representation.

Their approach is to partition the input vector into a prescribed number of equal-sized sub-vectors that are each quantized independently, using k-means and a constant number of bits per sub-vector, and then form the Cartesian product of each of the independently quantized components. Inter-point distances are estimated from the quantized values.

Among other things, they observe that it is possible to learn the sub-vector quantizers using a relatively small training set, even though there may be a large number of bits in the complete code. This is a crucial point given that it is often impossible to collect and analyze a sufficient number of training samples to span the quantized space, which often comprises hundreds of bits.

Similar to the product quantization approach of [7], Tuytelaars et al [15] proposed a lattice quantizer for SIFT descriptors, allocating a constant number of bits per descriptor component and show its effectiveness using uniform quantization and 2 bits per component.

Winder et al [17] explored several compression techniques for feature matching as part of a systematic exploration of the design space for local feature descriptors. They found that error rates were reduced through the use of PCA, dimensionality reduction, and uniform quantization using a constant number of bits per dimension.

In this paper, we propose a new quantization-based approach to nearest neighbor search inspired by well-established results in signal representation and compression. The new contribution, distinct from previous approaches, is that it combines transform coding, data-driven allocation of bits to components, and non-uniform minimum distortion quantization, to obtain a compact representation suitable for nearest neighbor search. Our approach is also distinct in that it is very general, making few assumptions of the data, applies to a broad range of metrics, and requires no parameters other than the number of bits. Experiments on real-world data, using a variety of metrics, validate the effectiveness of the approach.

3. The Retrieval Problem

Consider a finite set of points $X \subset \mathbb{R}^n$, $|X| = N$, drawn from the probability distribution $p(\mathbf{x})$ defined over $\mathbb{R}^n$. Point proximity is determined by a metric $d(\mathbf{x}, \mathbf{x}')$. The two fundamental proximity queries are *Radial*, namely to return the set of points within a given radius of the query, and *Nearest-k*.

Hash-based retrieval consists of a quantization step followed by lookup based on the quantized representation. That is, the quantization step identifies the unique partition containing $\mathbf{x}$ within a finite partitioning of $\mathbb{R}^n$. The index lookup returns all $\mathbf{x}' \in X$ contained within the given partition.

Hash-based lookup is effective for near-duplicate search, where we can rely on a hash collision even when the representation consists of a large number of bits. However, for proximity searches, such as Radial and Nearest-k, hash lookup becomes ineffective due to the sparseness of the code space.

What is needed for Radial and Nearest-k searches is an efficient quantization scheme that preserves distances. In other words, we wish to design a quantizer such that the induced partitioning of the input space is optimal for proximity search.

3.1. Minimum Distortion Quantization

Quantization in general has been the subject of prolonged study and has a rich and extensive literature [3]. Any quantizer, $q : \mathbb{R}^n \to Z$, where $Z = \{0, 1, \ldots, m-1\}$, is characterized by the partition it induces on the input space,

$$Q(z) = \{\mathbf{x} : q(\mathbf{x}) = z\}, \tag{1}$$

for $z \in Z$, and the codebook values associated with each z, $\mathbf{c}(z) \in \mathbb{R}^n$.

The quality of a given quantizer is measured in terms of its average distortion,

$$D = E[d(\mathbf{x}, \mathbf{c}(q(\mathbf{x})))], \tag{2}$$

where the distortion function d can take on a variety of forms. For retrieval, the appropriate distortion function to be minimized is simply the metric $d(\mathbf{x}, \mathbf{x}')$. In fact, application of the triangle inequality yields

$$E[|d(\mathbf{x}, \mathbf{x}') - d(\mathbf{x}, \mathbf{c}(q(\mathbf{x}')))|] \le D. \tag{3}$$

Therefore, D is an upper bound on the expected error in estimating inter-point distances when one of the two points is approximated by its quantized codebook value. Consequently, a quantizer that minimizes D for a fixed m is expected to be effective from the standpoint of retrieval.

A quantizer that minimizes D subject to the underlying distribution $p(\mathbf{x})$ is characterized by the following properties:

1. $Q(z) = \{\mathbf{x} : d(\mathbf{x}, \mathbf{c}(z)) \le d(\mathbf{x}, \mathbf{c}(z')), \forall z' \in Z\}$, and
2. $\mathbf{c}(z) = \arg\min_{\mathbf{x}'} E_{\mathbf{x}}[d(\mathbf{x}, \mathbf{x}') | \mathbf{x} \in Q(z)]$.

That is, the quantization cells consist of points no further from the cell's code vector than from any other code vector, and that the code vector for a given cell is its centroid. This classic result, first developed by Lloyd [10] and independently by Max [12] for 1D signals, and later extended to vector quantization by Linde, et al [9], is the basis for the k-means algorithm.

3.2. Product Quantization

In 1D, design of a quantizer to satisfy the minimum distortion criterion is well understood [3]. But in higher dimensions difficulties arise. For instance, it can be challenging to obtain a sufficient sampling to characterize $p(\mathbf{x})$. Additionally, vector quantization is computationally expensive in high dimensions.

In the rare case that $p(\mathbf{x})$ is independent in its components, and the metric is of the form $d(\mathbf{x}, \mathbf{x}') = \sum_i d_i(x_i, x_i')$, where x_i are the components of $\mathbf{x}$, we can obtain a minimum distortion quantizer by forming the Cartesian product of the independently quantized components. That is, let

$$q(\mathbf{x}) = (q_1(x_1), q_2(x_2), \ldots, q_n(x_n)) = \mathbf{z}, \tag{4}$$

where $\mathbf{z} = (z_1, z_2, ..., z_n)$, $z_i = q_i(x_i)$.

The quantizer design problem reduces to a set of n independent readily solved 1D problems: each q_i is designed independently to minimize the expected distortion $D_i = E[d_i(x_i, c_i(q_i(x_i)))]$. It is straightforward to show that $D = \sum_i D_i$. Therefore minimizing each D_i independently minimizes D.

3.3. Bit Allocation

The minimum distortion criterion is sufficient to design a product quantizer if the number of distinct quantization levels per component is known. Determining the number of levels per component is known as *bit allocation* [2, Ch. 8]. Optimal bit allocation amounts to minimizing

$$D(\mathbf{b}) = \sum_i D_i(b_i), \tag{5}$$

such that $\sum_i b_i = \log_2 m$, and where $\mathbf{b} = (b_1, b_2, ...)$ is the vector of per-component bit allocations.

Solution to this optimization problem for general distributions and distortion functions requires computationally prohibitive numerical search. A reasonable approximation, which turns out to be effective in practice, is to assume that each component is identically distributed after normalizing the variance, and that the per-component distortion functions are identical. In this case, the optimal allocation is achieved when

$$b_i \sim \log_2 \sigma_i, \tag{6}$$

where σ_i is the standard deviation of the i-th component [2]. Therefore, we can simply estimate σ_i from the training data and allocate bits to each component proportionally.

In practice, we prefer each b_i to be integer-valued so that the components $q_i(x_i)$ can be concatenated to encode $q(\mathbf{x})$ as a contiguous bit vector. We can ensure the overall bit budget is met while proportionally allocating an integer number of bits to each component through the sequential distribution procedure listed in Algorithm 1.

Algorithm 1 Distribute b bits among quantizer components.

```
Initialize $H(i) \leftarrow \log_2 \sigma_i$, $B(i) \leftarrow 0$.
for $j = 1$ to $b$ do
    $i \leftarrow \arg\max H(i)$.
    $B(i) \leftarrow B(i) + 1$.
    $H(i) \leftarrow H(i) - 1$.
end for
```

After the bit allocation procedure, it may be that some components are allocated no bits at all. The effect is dimensionality reduction since such components are omitted from $\mathbf{z}$. Therefore, we have a principled method to select a subset of dimensions, and simultaneously allocate bits to the remaining dimensions, given a fixed overall bit budget, while minimizing D.

Note that spectral hashing [16] arrives at a similar bit distribution scheme through a very different approach. The geometric intuition is that the resulting quantization cells should be roughly isotropic with respect to the distortion function. Therefore, components with larger standard deviations require a proportionally larger number of quantization levels.

3.4. Transform Coding

The product quantizer depends critically on the unrealistic assumption that the components of $\mathbf{x}$ are statistically independent. In vector quantization, this difficulty is addressed through *transform coding*, which seeks a (typically linear) transformation to reduce statistical dependence among the components [2].

Classically, this is done through principal component analysis (PCA), although many other alternatives exist. Specifically, we compute the eigenvectors and eigenvalues of the training sample covariance. The matrix of eigenvectors constitutes a unitary transformation U that is applied to all points prior to quantization. Then, the product quantizer is designed for points $\mathbf{y} = U\mathbf{x}$. Bit allocation and consequent dimensionality reduction, as described in the previous section, is based on the corresponding eigenvalues.

In summary the quantizer design method is as follows: (1) determine the PCA transformation from the training set, including removing the mean; (2) run the bit allocation procedure based on the PCA eigenvalues, and drop any com-

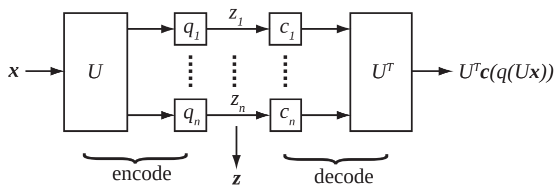

Figure 1. Encoding and decoding flow for the transform coding system.

ponents that receive zero bits; (3) for each remaining component, train a 1D distortion minimizing quantizer to obtain the quantization functions q_i and centroids c_i. The 1D distortion function D_i is the absolute difference in component values. Encoding of a new point entails PCA projection followed by quantization of the components. Reconstruction of a centroid in the input space from quantized point entails constructing the centroid $\mathbf{c} = (c_1, c_2, \ldots, c_r)$ in the transform space, and then back projecting it with U^T and adding back the mean. The encoding and reconstruction process is depicted graphically in Figure 1.

Since PCA does not guarantee statistical independence of the transformed components, the product quantizer is not in general optimal with respect to distance distortion. Additionally, with the exception of the squared Euclidean metric, quantizer distortion may not be expressible as the sum of the per-component distortion functions D_i. However, the suboptimality of the quantizer is offset by its simplicity in training and low computational cost. Additionally, experiments using real world data indicate that the transform coder outperforms competing methods in a broad set of situations, including those with non-Euclidean metrics (see Section 4).

To examine the information efficiency of the transform coding system by way of example, we consider the 128D SIFT descriptor compressed to 64 bits. The upper portion of Figure 2 depicts the bit allocation to each of the 45 highest energy principal components. The remaining components receive zero bits and are consequently dropped from the code. The lower portion of Figure 2 depicts the mutual information between each pair of components after quantization. This is an indication of second-order statistical dependencies that remain after the PCA projection. All of the off-diagonal terms are small; the largest off-diagonal term is 0.55 bits in this case. Additionally, the sum of the diagonal terms is 62.2 bits, indicating that little is to be gained by further compression, such as through entropy coding.

3.5. Distance Estimation

Returning to the retrieval problem, both Radial and Nearest-k queries require estimation of $d(\mathbf{x}, \mathbf{x}')$ for each retrieved point $\mathbf{x}' \in X$. In practice, it is typically too expensive to physically retrieve the points or to evaluate the exact distances, which is the reason we seek compact codes in the first place. In these cases, the retrieved points can be ranked based on the distances from the query point to the centroids for each retrieved point.

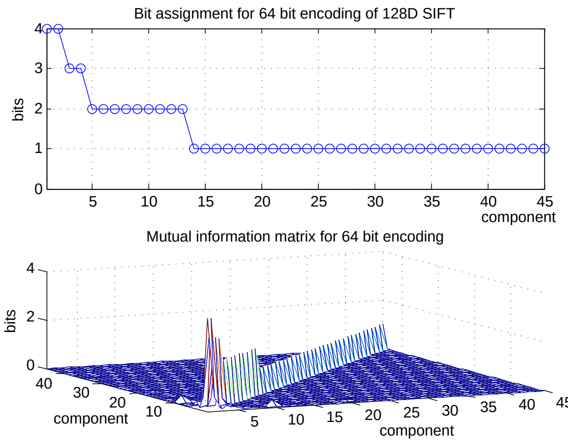

Figure 2. Bit assignment for 64 bit encoding of 128D SIFT descriptor. The upper plot depicts the number of bits assigned to each of the top 45 principal components. The lower plot depicts the mutual information between components of the resulting quantized descriptors.

The centroid for a particular quantization cell can be constructed by inverting the projection as depicted in Figure 1. We denote the resulting estimated distance as d_{IA}, referring to the fact that it is evaluated in the input space:

$$d_{IA}(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, U^T \mathbf{c}(q(U\mathbf{x}'))) \tag{7}$$

The function d_{IA} corresponds to the "asymmetric distance computation" formulated in [7].

Distance between centroids, measured in the input space, is

$$d_{IS}(\mathbf{x}, \mathbf{x}') = d(U^T \mathbf{c}(q(U\mathbf{x})), U^T \mathbf{c}(q(U\mathbf{x}'))) \tag{8}$$

corresponds to the "symmetric distance computation" formulated in [7].

Since d_{IS} includes quantization noise for both points, rather than just one, it is a poorer estimate of d than d_{IA}, but has the advantage of being static for a given quantizer, independent of the query, and consequently can be precomputed. Unfortunately, for a large number of bits, it is impractical to enumerate and store pairwise distances between all centroids. However, we escape this problem by computing distances in the transform domain instead of the input domain. Specifically, let

$$d_{TA}(\mathbf{x}, \mathbf{x}') = d(U\mathbf{x}, \mathbf{c}(q(U\mathbf{x}'))), \tag{9}$$

and

$$d_{TS}(\mathbf{x}, \mathbf{x}') = d(\mathbf{c}(q(U\mathbf{x})), \mathbf{c}(q(U\mathbf{x}'))), \tag{10}$$

Computing the distances in the transform domain introduces additional error related to the amount of signal energy discarded in the PCA projection. Also, evaluating distance in the transform domain only makes sense for metrics

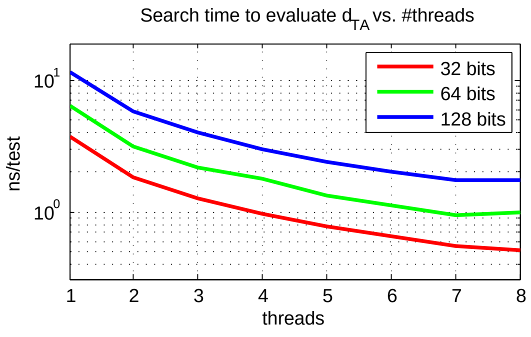

Figure 3. Exhaustive search time (in nanoseconds per tested point) to evaluate d_{TA} on quantized SIFT descriptors vs. number of threads at varying bit rates. Search times are the average of 10 passes over a 100 MB dataset.

that naturally extend to smaller dimensions, such as the Euclidean metric.

3.6. Optimized Linear Search

Evaluation of d_{TA} can be very fast using 1D lookup tables constructed at query time, which is appropriate in the large-scale retrieval setting where the cost of lookup table construction is small compared to the cost of search. Evaluation of d_{TS} can be implemented using static (query independent) 2D lookup tables, and therefore can be practical in an image matching setting.

Figure 3 plots the search times obtained by exhaustively evaluating d_{TA} on a large set of quantized SIFT descriptors at several bit rates using the squared Euclidean metric. In each case, the code is decomposed into a tuple of 8 bit words. Components of the quantized descriptor are permuted as needed to ensure that no component straddles a word boundary. Lookup tables are constructed for each word to store the partial distance determined by the components belonging to that word. For a code consisting of b total bits, we require $\lceil b/8 \rceil$ 256-entry lookup tables.

Distance evaluation for a single point consists of summing the lookup table values for the given quantized point, and appending the point index to a row in a preallocated 2D output buffer indexed by the quantized distance value. The number of rows in the output buffer determines the maximum search radius, and the number of columns limits the total number of points that are kept for a given quantized distance. After passing over the entire data set, the output buffer is scanned to extract the closest k indices.

The execution times in Figure 3 were obtained using a dual quad-core Intel Xeon X5550 CPU running at 2.67GHz. The linear scan is easily parallelizable; hence we see near-linear speed-up as the number of threads is increased. At 64 bits, we are able to average one point per nanosecond using 8 threads, enabling exhaustive search of one billion points in a second.

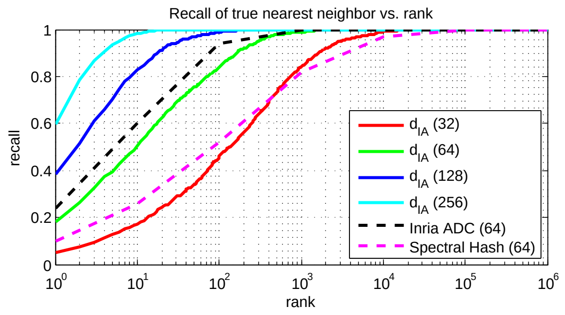

Figure 4. Retrieval results on INRIA Holidays 1 million SIFT descriptor dataset using several different encoding schemes. "d_{IA} (n)" denotes the transform code described here using n bits and the distance estimator d_{IA}. "Inria ADC (64)" is the 64 bit vector quantization scheme proposed in [7]. "Spectral Hash (64)" is the 64 bit spectral hash.

A variety of techniques exist for asymptotically sublinear search using hierarchical structures of one sort or another. The advantage of linear search is that it is ideally suited to modern system architectures optimized for high memory locality and streaming data.

4. Results

Development of the transform coder depends on two important assumptions that may not be valid in practice. One is that PCA is effective at reducing dependencies among the components, and the second is that the 1D minimum distortion quantizers are effective at limiting distance estimation error. The former assumption depends on the distribution $p(\mathbf{x})$, while the latter assumption depends primarily on the form of the metric.

In this section, we test the performance of the proposed method on a diverse range of tasks to validate if it is in fact effective in practice. In each experiment, the training and encoding algorithms are identical, parameterized only by the desired number of bits, and the applicable metric. The method of search varies, depending on the task.

4.1. Large-scale Retrieval

To test the effectiveness of our proposed method for large-scale retrieval, we employ the INRIA Holidays dataset,[1] consisting of 128D SIFT descriptors divided among a training set, search set, and query set. The training set was separately collected from a random sampling of Flickr images. The search set contains 1 million points and the query set contains 10 thousand. Descriptor similarity is the Euclidean metric.

Figure 4 depicts "Recall @ R" (the probability that the nearest point to a query is among the most relevant R points

[1] Available at http://lear.inrialpes.fr/˜jegou/data.php.

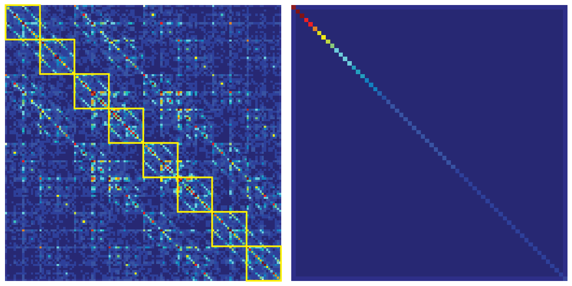

Figure 5. Covariance matrix of 128D SIFT training sample before (left) and after (right) PCA transformation. The yellow squares highlight the covariances of the vector quantized sub-vectors of [7].

returned by a given encoding scheme) for a range of values of R, averaged over a large number of queries. These results are directly comparable to those published in [7], which have been transferred to this plot for comparison purposes.

At 64 bits, the transform code performs significantly better than spectral hashing [16], achieving equal retrieval rates at roughly an order of magnitude smaller R. The accuracy of the 64 bit transform code is slightly lower than that of [7]. In their case, the structure of the SIFT descriptor is exploited to attain good vector quantization efficiency. The transform code makes no assumptions about the structure of the data, and instead learns it through PCA.

Figure 5 depicts the training sample covariance before and after PCA. The method of [7] vector quantizes each of the 8 16D sub-vectors, thus representing the 8 16x16 blocks (depicted as yellow squares in the figure) along the main diagonal using a constant number of bits per sub-vector using vector quantization. In contrast, the transform coder encodes each of the components independently after most of the statistical dependencies between components have been eliminated through PCA. It also allocates bits to the components proportional to the component energies.

In comparison to [7], the transform code is more simple and efficient to train and to encode descriptors. Additionally, linear search is faster than that reported in [7], although these numbers are difficult to compare. The computational complexity of transform code training and encoding grows slowly as the number of bits is increased, while with the product of vector quantizers complexity growth is exponential in the number of bits, unless the descriptor is further decomposed into smaller units which may be less advantageous with respect to the structure of the data. Figure 4 confirms that the performance of the transform coding method improves continuously with the number of bits.

In summary, the two methods are similar, with the method of [7] having a slight edge in accuracy on SIFT data, while the method proposed here is simpler, faster, and arguably more general because it adapts to the data as part of the training process, and applies to non-Euclidean metrics.

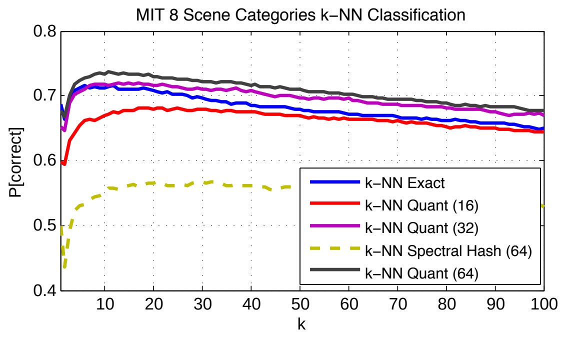

Figure 6. k-Nearest neighbor classification on MIT scene categories data set using the 960D GIST descriptor. "k-NN Exact" uses the exact Euclidean distance on the uncompressed descriptors. "k-NN Quant (n)" denotes the classifier based on the n bit transform code. "k-NN Spectral Hash (64)" denotes the classifier based on the 64 bit spectral hash with Hamming metric.

4.2. k-Nearest Neighbor Classification

A major challenge for scene category and object category classification is to develop techniques that remain practical as the scale increases to thousands or tens of thousands of categories. If we are able to do fast nearest neighbor search, then the k-nearest neighbor (k-NN) classifier is one technique that has the potential to do so.

The idea of using compact codes for recognition was explored in Torralba et al [14], where they applied their learned compact codes and Hamming-space search. Since the transform coder limits distance estimation error, we expect the transform coder to perform better at equal bit rates.

To test the effectiveness of transform coding and quantization in the context of k-NN, the MIT scene category dataset[2] was evaluated. The dataset consists of 2,688 images distributed among 8 scene categories. The training data consist of 100 randomly selected images from each category. The remaining images constitute the test set. Nearest neighbor search is based on the Euclidean metric applied to the 960D GIST descriptor computed for each image.

In Figure 6, the probability of correct classification using the k-NN criterion (namely, vote among the labels of the k nearest neighbors) is plotted as a function of k, at varying levels of quantization. The reported values are the average over 10 trials, with each trial having a different randomly selected training set. In comparison, classification rates using spectral hash [16] are significantly lower at equal bit rates.

The results indicate that a 32 bit representation of the GIST descriptor is sufficient to exceed the performance of the exact k-nearest neighbor on this data set. The increase in performance is somewhat surprising, but is likely due to the dimensionality reduction intrinsic to the quantization process which serves to combat over-fitting. Even at 16 bits,

[2] Available at http://people.csail.mit.edu/torralba/code/spatialenvelope.

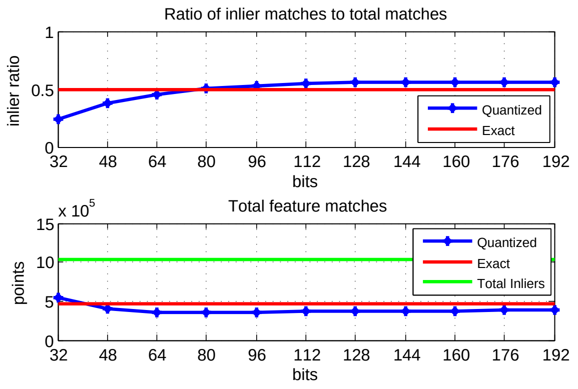

Figure 7. Effectiveness of quantized SIFT descriptor for image matching. The top plot shows the fraction of matched inliers using the quantized representation at varying bit rates. The bottom plot shows the total number of matched points using the quantized representation. "Exact" refers to the inlier ratio and number of matches obtained using the uncompressed descriptor.

the performance is close to that of the exact method, and significantly better than the 64 bit spectral hash.

4.3. Local Feature Matching

Local feature matching refers to the process of forming the correspondence between two images using local feature descriptors, for instance using Lowe's SIFT algorithm [11] to identify candidate corresponding feature point pairs, followed by RANSAC to determine a geometrically consistent subset of the candidate pairs, identified as inliers. It is natural to ask whether the proposed quantized representation of the local feature descriptor is adequate to this task. If so, then we have the opportunity to compress the descriptor, without loss of expressiveness, for bandwidth-limited applications, and to accelerate the matching process.

To test the idea, we collected panorama image sets and registered them to obtain a ground truth homography between each overlapping image pair. The images are of varying resolution and subject matter, including natural and man-made settings. In total, the test set comprises 891 registered image pairs. Feature points were obtained from the images using Lowe's DOG detector. Each feature point is represented using the standard 128D SIFT descriptor.[3]

The experiment is to encode the SIFT descriptors at varying bit rates and measure the effectiveness of the matching process using the compressed representation in comparison to using the uncompressed representation. The methodology is to consider each pair of images as in the role of "source" and "target" for matching. For each source/target image pair, define the "true inliers" as the set of feature pairs (f_s, f_t) such that f_s is the closest source feature point to the target feature point f_t after it is mapped to the source image under the known homography, and such that the distance between the two feature points in the source image is less than a fixed radius (5 pixels). We then use Lowe's distance ratio criterion to identify candidate matches based on the exact descriptor values and the Euclidean metric. (Lowe's criterion is that the ratio of the distance in descriptor space of the closest descriptor to the second closest be less than a given fixed threshold — 0.8 for these experiments). The fraction of true inlier pairs that are in the matched set is the "inlier ratio" and is related to the likelihood that the images can be registered using RANSAC. We then compress the descriptors using a transform coder that was trained on a disjoint set of images and apply the same distance ratio criterion based on the d_{TS} distance defined in (10).

The foregoing procedure was done for each pair of images and the aggregate results appear in Figure 7. The upper plot is of the mean inlier ratio over a range of bit rates, while the lower plot is of the total number of matches over the same range of bit rates. At 80 bits, or $12.8 : 1$ compression, the quantized descriptor is equally effective at identifying inlier matches as the uncompressed descriptor. The total number of matches using the quantized descriptor is less than that using the exact descriptor due to systematic bias in the quantized distance estimate. (Quantized distances underestimate the exact distance.) Nevertheless, we can conclude that the quantized descriptor of 80 bits is sufficient for local feature matching in panoramas.

As mentioned above, evaluation of d_{TS} can be extremely fast using static 2D lookup tables. Also, the compression obtained by use of the quantized descriptor can significantly improve performance in circumstances where the descriptors must be transmitted over a network.

4.4. Spatial Pyramid Bag-of-Words Retrieval

The spatial pyramid bag-of-words scene representation proposed by Lazebnik et al [8] has been shown to be effective for scene category classification, and scene similarity retrieval. However, it is problematic to use in a large scale because the descriptors have high dimension (typically thousands of components), but are not sufficiently sparse for sparse methods, such as min-hash or inverted files, to be effective.

To test whether the transform code is effective in this setting, we computed spatial pyramid bag of words descriptors for each of the images in the MIT Indoor Scene Category dataset.[4] Specifically, for each image, we collected dense SIFT features in a grid pattern, quantized the feature descriptors into a vocabulary of 200 visual words, and then formed a three level spatial pyramid of histograms, resulting in a descriptor of 4,200 dimensions. (The vocabulary was learnt on a disjoint image set.) On average, 25% of the

[3]The dataset is available at http://www.adobe.com/go/datasets.

[4]Available at http://web.mit.edu/torralba/www/indoor.html.

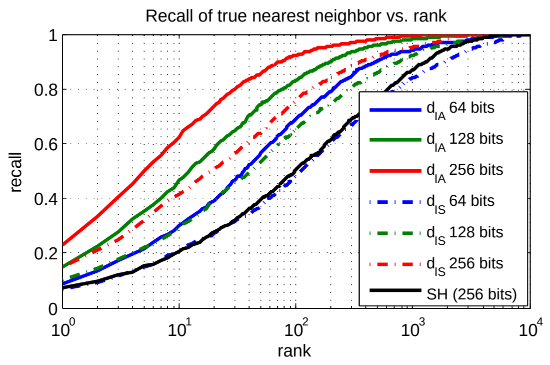

Figure 8. Effectiveness of quantized descriptor at several bit rates for image similarity search on spatial pyramid bag of words representation using weighted histogram intersection metric. For comparison, results based on spectral hashing ("SH") are also shown.

descriptor components was non-zero. The similarity metric was the TFIDF-weighted histogram intersection metric.

The experiment was to test the effectiveness of the transform coder on this very high-dimensional descriptor and non-Euclidean metric in a retrieval setting. Specifically, no attempt was made to classify the images. The entire dataset consists of $15,620$ images. In each trial, $5,000$ were randomly selected for training, $10,000$ for the search set, and the remainder formed the query set.

Figure 8 depicts "Recall @ R" as defined in Section 4.1 at a range of bit rates, and using both the d_{IA} and d_{IS} metrics. (For comparison purposes, spectral hashing [16] was similarly evaluated, although we do not expect good performance because it is ill-suited to the histogram intersection metric.)

The results indicate that with a 256 bit code, or roughly $100 : 1$ compression, we can exceed 90% recall while retrieving only 1% of the data. Because of the non-Euclidean metric, it is necessary to evaluate distances in the input space which is more expensive than operating in the transform space. However, the most expensive part, reconstruction of the quantized points (evaluation of $U^T\mathbf{c}(q(U\mathbf{x}))$), can be accelerated through the use of lookup tables, not unlike the procedure described in Section 3.6. As expected, d_{IA} produces more accurate results than d_{IS} due to d_{IA} having less quantization noise.

5. Conclusions

We have presented a transform coding and quantization approach to high-dimensional approximate nearest neighbor search. The approach consists of PCA, followed by non-uniform data-driven bit allocation, followed by distortion-minimizing non-uniform product quantization. Training is extremely simple and fast, dominated by the eigenvector analysis. Encoding is even faster, consisting of a linear projection followed by a set of 1D quantizers.

Despite the simplicity of the approach, we have demonstrated its effectiveness in a range of settings, including large scale retrieval, scene classification, feature matching, and image similarity using a non-Euclidean metric. The resulting compressed code enables extremely fast, trivially parallelizable, large-scale search. The algorithm is parameterized only by the number of bits and the applicable metric.

Future work includes investigating alternatives to PCA, extending the range of applicable metrics, including kernel-based representations, exploring further applications, and accelerating search using the GPU.

References

[1] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *J. of the ACM*, 45:891–923, 1998.

[2] A. Gersho and R. Gray. *Vector quantization and signal compression*. Kluwer, 1991.

[3] R. Gray and D. Neuhoff. Quantization. *IEEE Trans. on Inf. Th.*, 44(6):2325–2383, 1998.

[4] P. Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*. CRC Press, 2004.

[5] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008.

[6] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *ICCV*, 2009.

[7] H. Jégou, M. Douze, and C. Schmid. Searching with quantization: approximate nearest neighbor search using short codes and distance estimators. Technical Report RR-7020, INRIA, August 2009.

[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.

[9] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Comm.*, 28(1):84–95, 1980.

[10] S. Lloyd. Least squares quantization in PCM. *IEEE Trans. on Inf. Th.*, 28(2):129–137, 1982 (Reprint of 1957 Bell Labs Technical Report).

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

[12] J. Max. Quantizing for minimum distortion. *IRE Trans. on Inf. Th.*, 6(1):7–12, 1960.

[13] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest neighbor methods in learning and vision: theory and practice*. MIT Press, 2006.

[14] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[15] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, 2009.

[16] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[17] S. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, 2009.