Compact Projection: Simple and Efficient Near Neighbor Search with Practical Memory Requirements

Kerui Min[1,2], Linjun Yang[2], John Wright[2], Lei Wu[2,3], Xian-Sheng Hua[2], Yi Ma[2]
[1]School of Computer Science, Fudan University
krmin@fudan.edu.cn
[2]Microsoft Research Asia
{linjury,jowrig,xshua,mayi}@microsoft.com
[3]MOE-MS Keynote Lab of MCC, University of Science and Technology of China
wulei@live.com

Abstract

Image similarity search is a fundamental problem in computer vision. Efficient similarity search across large image databases depends critically on the availability of compact image representations and good data structures for indexing them. Numerous approaches to the problem of generating and indexing image codes have been presented in the literature, but existing schemes generally lack explicit estimates of the number of bits needed to effectively index a given large image database. We present a very simple algorithm for generating compact binary representations of imagery data, based on random projections. Our analysis gives the first explicit bound on the number of bits needed to effectively solve the indexing problem. When applied to real image search tasks, these theoretical improvements translate into practical performance gains: experimental results show that the new method, while using significantly less memory, is several times faster than existing alternatives.

1. Introduction

The growing popularity of online image and video sharing sites has made efficiently indexing and searching large image databases an extremely active area of research in recent years [5]. Much of this research has focused on the fundamental problem of retrieving images that are as similar as possible to a given query image. This *similarity search* is a key component of many commercial image search engines, helping to organize results and remove near duplicates. Current state-of-the-art systems for similarity search generally built around approximately invariant image descriptors such as SIFT [10] or GIST [12]. These features are quantized and their statistics are used to efficiently index the image database, using an inverted file system (see Figure 1). This methodology finds application throughout the computer vision literature, including areas that are not traditionally considered part of image search, such as 3D reconstruction [20] and face recognition [21].

The practicality of such image retrieval systems depends critically on two factors: the quality of the image representation employed, and the efficiency of the indexing system. Unfortunately, these two factors are often in tension. Research in image descriptors generally suggests that better discrimination can be achieved with higher dimensional feature descriptors. However, finding nearest neighbors in the high-dimensional feature space is impractical for web-scale image databases. The naive nearest neighbor algorithm suffers from the "curse of dimensionality", as do classical alternatives such as KD-trees [13] and its variants [6, 11], R-trees [7] etc.

The difficulty of exact nearest neighbor search has led to the development of many alternative strategies, with varying levels of practicality and theoretical optimality. Weber et al. [18] proposed a vector approximation scheme to accelerate the sequential scan. Wang et al. [17] proposed a binary hash coding (BHC), which projected images from a high dimensional feature space into a low dimensional hash code space and then filtered similar images by nearest neighbor search. Weiss et al. [19] used a spectral graph based method to find a best hashing code space in which Hamming distance is correlated with the semantic similarity. Torralba et al. [15] proposed to thumbnail the images into small code to speed up the similarity search in a large database. While many of the above techniques demonstrate excellent performance on specific image retrieval tasks and databases, they generally lack theoretical guarantees or guidelines as to the number of bits needed to achieve a given level of performance. This lack of foundation makes their application to

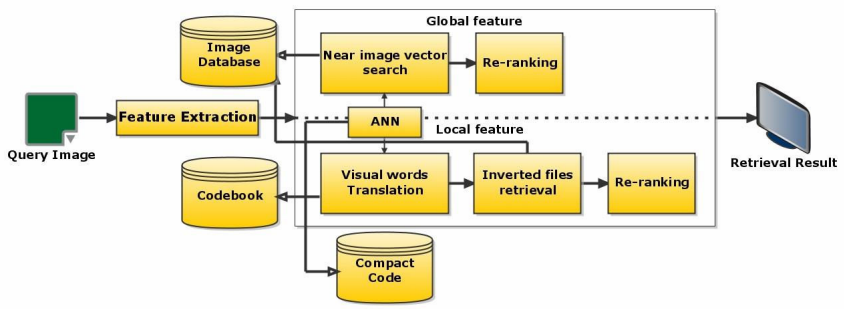

Figure 1. The general framework of a similar image retrieval system

new databases mostly a matter of trial-and-error.

On the other hand, researchers in the theoretical computer science community have developed approximate nearest neighbor algorithms with excellent performance guarantees, in some cases nearly optimal. The current state-of-the-art in this area is arguably the locality sensitive hashing (LSH) approach of [2, 4], which exploits the distance preserving properties of random projections. However, in spite of their appealing properties, there are practical drawbacks to this and related algorithms, most importantly intensive memory requirements that hinder their application to real image search tasks. This difficulty manifests itself in theory as well as practice: state-of-the-art schemes lack an explicit bound on the number of bits needed to effectively index a given set of data.

In this paper, we study a very simple algorithm for generating compact binary representations of a given image database. Like LSH, our algorithm is based on random projections. However, it achieves much better space complexity. We prove an explicit bound on the number of bits needed to encode the data with a certain guaranteed retrieval performance. Experimental results show that our algorithm, while using significantly less memory, is several times faster than existing alternatives.

2. Problem Statement and Algorithm

Our algorithm takes as its input a large set of high-dimensional vectors $X = \boldsymbol{x}_1 \dots \boldsymbol{x}_n \in \mathbb{R}^d$. In the context of image retrieval, these could be feature descriptors representing n images. Given a query vector $\boldsymbol{q} \in \mathbb{R}^d$, the core retrieval task is to efficiently return a vector $\boldsymbol{x}^* \in X$ that is as similar as possible $\boldsymbol{q}$: the distance $d(\boldsymbol{q}, \boldsymbol{x}^*)$ is as small as possible. The correct distance $d(\cdot, \cdot)$ may vary depending on application and dataset. However, if the image features are well-chosen, the ℓ^p norms

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p = \left(\sum_{i=1}^{d} |\boldsymbol{x}(i) - \boldsymbol{y}(i)|^p \right)^{1/p}$$

have been observed to perform well for retrieval tasks [10, 16]. In this paper, we focus on the Euclidean case $p = 2$.

The most straightforward approach to this problem is to simply scan through the database and return the point $\boldsymbol{x}^* \in X$ that minimizes $d(\boldsymbol{q}, \boldsymbol{x}_i)$, yielding an excellent neighbor: $d(\boldsymbol{q}, \boldsymbol{x}^*) = \min_{\boldsymbol{x} \in X} d(\boldsymbol{q}, \boldsymbol{x})$. However, this approach requires $O(dn)$ operations. In practical image retrieval scenarios, both n and d could be very large: web image databases such as Flickr contain millions of images or more, while for excellent retrieval performance is generally believed to require high-dimensional image representations (e.g., $d = 512$ dimensions for the GIST descriptor [12]). This unfavorable complexity rules out the trivial algorithm for all but the smallest image databases. Fortunately, in applications such as image search it is not necessary to return the point that is nearest to the query point; a set of well-chosen *near neighbors* may suffice. This gives the algorithm designer considerable freedom to trade off approximation error for search complexity. This problem can be formalized as follows:

Problem 1 (c-approximate near neighbor (c-NN)). *Given a set X of points in a d-dimensional space $\mathbb{R}^d$, devise a data structure, which for any query point $\boldsymbol{q} \in \mathbb{R}^d$ finds a point $\boldsymbol{x}^* \in X$ such that $d(\boldsymbol{q}, \boldsymbol{x}^*) \leq c \cdot \min_{\boldsymbol{x} \in X} d(\boldsymbol{q}, \boldsymbol{x})$.*

This problem has received a great deal of attention in the theoretical computer science community (see [14] and the references therein). Some of the simplest and best-performing algorithms for solving it rely on the fact that in high dimensional spaces, random projections preserve distances with very high probability. More precisely, if we sample a random vector $\boldsymbol{r} \in \mathbb{R}^d$ with components iid $\mathcal{N}(0, 1)$, then $\mathrm{var}(\boldsymbol{r} \cdot (\boldsymbol{x} - \boldsymbol{y})) \sim \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$. *Locality Sensitive Hashing* (LSH) builds on this property to give a state-of-the-art approximate nearest neighbor scheme [2]. However, that algorithm relies on an amplification technique that essentially trades off storage space for query time. The result is a scheme that is extremely memory intensive, requiring superlinear space, which limits its practical applicability to large image databases.

Algorithm 1 Compact Projection Generation

Input: Data samples $\boldsymbol{x}_1 \dots \boldsymbol{x}_n \in \mathbb{R}^d$.
Output: Compact projections $\boldsymbol{y}_1 \dots \boldsymbol{y}_n \in \{0,1\}^k$.
Generate $R \in \mathbb{R}^{k \times d}$ with independent normal entries $R_{ij} \sim N(0,1)$.
for each $i = 1 \dots n$ **do**
  Compute $R\boldsymbol{x}_i$
  Set $\boldsymbol{y}_i = \sigma(R\boldsymbol{x}_i)$.
end for

In this paper, we show that for real data arising in vision problems it is possible to rely on a much simpler hashing scheme, that alleviates the memory bottleneck associated with LSH, and comes with an explicit theoretical guarantee on the number of bits of memory needed to solve the c-approximate nearest neighbor problems. We represent each of the given data points $\boldsymbol{x}_i$ using a k-bit binary code $\boldsymbol{y}_i \in \{0,1\}^k$. This code is obtained by randomly projecting $\boldsymbol{x}_i$ into a k-dimensional space, and then quantizing the projections according to their signs. More precisely, for $x \in \mathbb{R}$, let

$$\sigma(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}.$$

We generate our compact code by sampling a $k \times d$ matrix $R \in \mathbb{R}^{k \times d}$, with entries independently from a standard normal distribution, and setting

$$\boldsymbol{y}_i = \sigma(R\boldsymbol{x}_i) \in \{0,1\}^k. \tag{1}$$

In Section 3, we prove that the number of bits k needed to guarantee good retrieval performance is quite small: $k = O(\log n)$.

Because of this low memory requirement, at query time we can efficiently compare the compact code for the given query image $\boldsymbol{q}$ to the compact codes for each of the database images $\boldsymbol{x}_i$. We set $\boldsymbol{y}_q = \sigma(R\boldsymbol{q}) \in \{0,1\}^k$, and select the $\mathcal{T}$ points $\boldsymbol{x}_i$ that minimize the Hamming distance $\|\boldsymbol{y}_q - \boldsymbol{y}_i\|_0$. As we will show, the number $\mathcal{T}$ can be chosen to be significantly smaller than the total number of images n. Amongst this much smaller set of candidates, we can then return the point $\boldsymbol{x}^*$ with minimum distance to $\boldsymbol{q}$, measured in the original space.

These simple encoding and query procedures are summarized as Algorithms 1 and 2, respectively. We call the output of the encoding scheme a *Compact Projection*, because it efficiently represents the input image using only a few bits. In the next section, we will prove that for datasets satisfying a certain *separability* condition, this algorithm guarantees to solve the approximate nearest neighbor problem with high probability. In section 5, we will then show that these theoretical considerations lead to more efficient image retrieval in practical scenarios.

Algorithm 2 Query Procedure

Input: Query point $\boldsymbol{q} \in \mathbb{R}^d$, compact projections $\boldsymbol{y}_1 \dots \boldsymbol{y}_n$ for the n database images.
Generate the code $\boldsymbol{y}_q = \sigma(R\boldsymbol{q})$.
for each $i = 1 \dots n$ **do**
  Compute the Hamming distance $\|\boldsymbol{y}_q - \boldsymbol{y}_i\|_0$.
end for
Select the points $\tilde{X} \subset X$ with whose codes have $\mathcal{T} = O\left(n^l\right)$ $(0 < l < 1)$ smallest Hamming distance from $\boldsymbol{y}_q$.

Return $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \tilde{X}} d(\boldsymbol{x}, \boldsymbol{q})$.

3. Correctness and Computational Complexity

The random projection used in Algorithm 1 naturally discriminates between near and far neighbors, as shown by the following lemma:

Lemma 1. *([8]) For any fixed unit vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, if we draw $\boldsymbol{z} \sim_{iid} \mathcal{N}(0,1)$, then $\mathbb{P}[\operatorname{sign}(\boldsymbol{x} \cdot \boldsymbol{z}) \neq \operatorname{sign}(\boldsymbol{y} \cdot \boldsymbol{z})] = \frac{1}{\pi} \arccos(\boldsymbol{x} \cdot \boldsymbol{y}) = \frac{1}{\pi}\theta(\boldsymbol{x}, \boldsymbol{y})$.*

Thus, if the angle $\theta(\boldsymbol{x}, \boldsymbol{y})$ is large, Algorithm 1 is very likely to produce different compact codes for these two vectors. For data that are reasonably spread in the high-dimensional space $\mathbb{R}^d$, only a small number $O(\log n)$ bits will be required to produce good codes. We formalize this spreadedness requirement in the following definition:

Definition 1 (Weak separability)**.** *Call a point set $X \subset \mathbb{R}^d$ Δ-weakly separable if for any query point $\boldsymbol{q} \in \mathbb{R}^d$,*

$$|\{i \mid \theta(\boldsymbol{q}, \boldsymbol{x}_i) \leq \Delta\}| = O(n^l).$$

Although it is defined in terms of arbitrary $\boldsymbol{q} \in \mathbb{R}^d$, the following lemma shows that it is sufficient to check this condition across only $\boldsymbol{q} \in X$:

Lemma 2. *If for every $\boldsymbol{x}_j \in X$, $|\{i \mid \theta(\boldsymbol{x}_j, \boldsymbol{x}_i) \leq 2\Delta\}| = O(n^l)$, then X is Δ-weakly separable.*

In Section 5, we use the above lemma to estimate the parameters $\mathcal{T}$ and l for real vision datasets. We will show that they are quite small over a wide range of Δ ($l \approx 0.4$ for SIFT), making weak separability a very relevant assumption for assessing the performance of hashing algorithms on visual data. Our main theoretical result shows that whenever this separability assumption holds, Compact Projection guarantees to give excellent hashing performance with a small number of bits, logarithmic in the size of the database:

Theorem 1. *Given the query $\boldsymbol{q}$, with probability of $1 - \delta$, c-approximate near neighbor problem can be solved on the Δ-weakly separable dataset with the target number of bits k in Algorithm 1 setting to $k = O\left(\frac{\ln(2/\delta) + \ln n}{(1 - 1/c)^2 \Delta}\right)$.*

Proof. Please see the appendix. □

This theoretical result leads to a practical approximate nearest neighbor scheme, with the following complexity guarantees:

Corollary 1. *Given the successful probability* $1-\delta$*, and the approximation parameter* c*, the time complexity of*

- *Construction:* $O\left(dn\log n\right)$.
- *Query:* $O(n+dn^{l})$.
- *Index space:* $O(n)$.

Note: here, we adopt the standard $(\log n)$-RAM computational model, in which arithmetic operations with $\log n$ bits can be performed in $O(1)$ time.

Proof. Since Theorem 1 gives the upper bound of the column of matrix R, the trivial matrix multiplication[1] yields the time complexity of the $O\left(\frac{nd(\ln(2/\delta)+\ln n)}{(1-1/c)^2\Delta}\right) \sim O(dn\log n)$. The $(\log n)$-RAM model allows us to compute $\log n$ bits arithmetic operation in $O(1)$ time. Thus $\|\boldsymbol{y}_q-\boldsymbol{y}_i\|_0$ in Algorithm 2 can be computed in constant time. The verification of the second stage of our algorithm require to check $\mathcal{T}=O(n^l)$ points, thus the total time complexity for a query is $O(n+dn^l)$. □

In the next two sections, we compare the performance of this scheme with existing results, both theoretically (Section 4) and empirically (Section 5).

4. Comparison with Previous Results

In this section, we compare our results in Theorem 1 and Corollary 1 to the best existing schemes for approximate nearest neighbor. As we will see, although many methods exploit the attractive properties of random projections, our result is the only one that provides explicit control of the number of bits needed to encode the data. Section 5 will show how these theoretical advantages lead to practical gains when indexing large image databases.

Random Projections. Much of the appeal of random projection techniques comes from the Johnson-Lindenstrauss lemma [9], which states that for an arbitrary set of n points X, a random projection $f:\mathbb{R}^d\to\mathbb{R}^k$ into a $k=O(\log n)$ dimensional space nearly preserves the pairwise distances between the points: with high probability

$$(1-\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \le \|f(\boldsymbol{x})-f(\boldsymbol{y})\|_2^2 \le (1+\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$$

for all pairs $\boldsymbol{x},\boldsymbol{y}\in X$. This phenomena is fairly flexible with respect to the distribution of f (see e.g., binary versions in [1]). However, a straightforward encoding of the projected points leads to a representation with $O(\log^2 n)$ bits per point. Conversely, our result shows that as long as the dataset satisfies a mild separability condition, far fewer bits suffice: $O(\log n)$ bits per point are enough to allow us to recover c-approximate near neighbors with high probability.

Random projection trees. The *random projection trees* analyzed by Dasgupta and Freund [3] split the high-dimensional ambient space $\mathbb{R}^d$ recursively along randomly chosen directions. This yields a quantization of the space that is adaptive to any low-dimensional structure in the dataset X: if X lie near some n-dimensional submanifold, then $O(\log(n/\delta))$ projections suffice to achieve an expected cell diameter δ [3].

However, this does not directly imply guarantees for near neighbor recovery. In contrast, our result gives an explicit bound on the number of bits needed to achieve a c-approximate nearest neighbor, and does not depend on any assumption on the dimensionality of the data, only on their separability. In fact, our results suggest that random projection trees may provide a good representation of a much broader range of datasets.

Locality-sensitive Hashing. As discussed above, Locality Sensitive Hashing is a state-of-the-art approximate nearest neighbor scheme with numerous successful applications [14]. It also uses quantized random projections to achieve the following time and space complexity:

- Construction: $O(dn^{1+\rho})$,
- Query: $O(dn^{\rho})$,
- Index space: $O(n^{1+\rho})$,

where $\rho>0$ depends on the approximation parameter c. The best known bound $\rho=1/c^2+O\left(\frac{\log\log n}{\log^{1/3} n}\right)$ was obtained in [2]. However, since the convergence of $O\left(\frac{\log\log n}{\log^{1/3} n}\right)$ is slow, the estimate $\rho\le 1/c$ given in [4] may be more relevant in practice.

CP has several advantages over LSH. Although at query time finding the $\mathcal{T}$ nearest projections requires $O(n)$ computation, the Hamming distance can be evaluated extremely efficiently. If l is small, say $l=0.4<1/c$, the computational cost for the second stage is also considerably lighter than that of LSH. Although d is usually considered constant, the value of d could be very large in practice, *i.e.* 512 dimensions for the Gist feature [12]. For high-dimensional data, for example in *proportional growth* $d=\Theta(n)$, CP has much lower time complexity than LSH.

[1] Fast algorithms for matrix multiplication can be employed to improve the theoretical construction time. However, trivial matrix multiplication is more practical, since it does not require the whole dataset X to be stored in main memory.

Another important issue is the space complexity. Notice that the space required by LSH is super-linear, *i.e.* $O(n^{1+\rho})$. In order to process $n = 10^6$ points with high success probability, LSH requires tens of gigabytes of memory, whereas CP needs only dozens of megabytes, allowing us to store the entire index in main memory. As we will see in the next section, this is a significant practical advantage.

5. Experiments on Large Image Databases

In this section, we study the performance of our compact projections on several real image retrieval tasks. We first corroborate our theoretical analysis by showing that CP indeed achieves highly accurate approximate nearest neighbor search using short codes. We further validate the *weak separability* assumption underlying our analysis by empirically determining the appropriate exponent l in Algorithm 2. Comparisons to two state-of-the-art methods, Binary Hash Codes [17] and Locality Sensitive Hashing [2] show that Compact Projection yields faster retrieval time with significantly smaller memory footprint. We show how these results can be used in a model image retrieval task, efficiently indexing $1M$ images from Flickr on a standard PC.

Datasets. We use three datasets to evaluate the effectiveness of Compact Projection. The *Oxford Building* dataset is a collection of around 5,000 images of 11 landmarks in Oxford. This yields approximately 15M SIFT descriptors. The *INRIA ANN* dataset is a publicly available benchmark for approximate near neighbor search. It contains over 1M SIFT features, with an additional 10,000 SIFT descriptors withheld as queries. Finally, to demonstrate our algorithm's applicability to real image retrieval tasks, we collect 1M images from Flickr, and represent each image with a single GIST feature vector.

Accurate retrieval with short codes. In this experiment, investigate how the length k of the compact projection affects the accuracy of near neighbor search. We choose a random set of 10^5 SIFT features from the *Oxford Building* dataset to serve as the database X, and another 10^4 features to serve as the query set. We consider compact projections of varying length $k = 64, 128, 256$ bits. For each, we vary the number of near codes $\mathcal{T}$ retained for exact search between 1 and $1,024$.

Figure 2 (top) plots the probability of successfully obtaining the exact nearest neighbor for each $\mathcal{T}, k$ combination, while Figure 2 bottom plots the probability of successfully obtaining a 1.5-near neighbor. Notice that with $k = 256$ we successfully obtain the exact nearest neighbor for 99.3% of the queries while retaining only $\mathcal{T} = 1,024$ candidates for the second stage of our algorithm, approximately 1% of the database. In the following experiments, we therefore fix $k = 256$ bits.

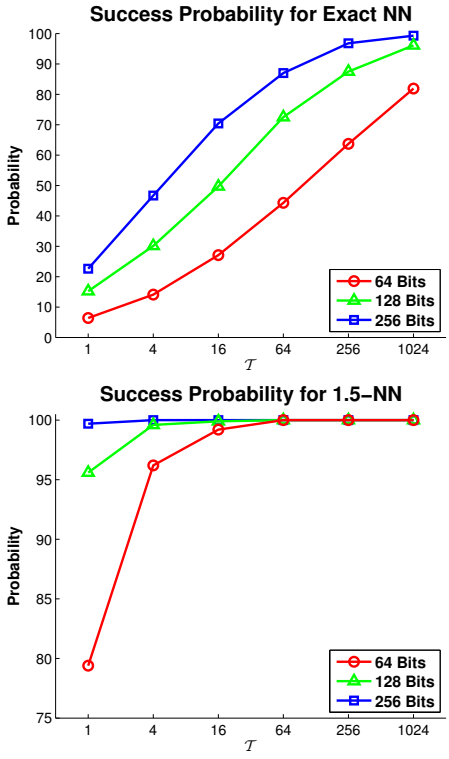

Figure 2. The probability of 64, 128 and 256 bits CP to find the near neighbors on the Oxford Building Dataset with respect to SIFT features. The x axis is the value of $\mathcal{T}$.

Real datasets are *weakly separable*. The previous result corroborates our analysis in Section 3: good near neighbors are obtained with relatively few bits per vector. In order to be confident that this performance generalizes, we next assess the validity of the *weak separability* assumption underlying Theorem 1. For this task, we use the database of 1M real images collected from Flickr.

We vary the size of the database and investigate the number of candidates $\mathcal{T}$ that must be retained to achieve a fixed probability p of recovering the exact nearest neighbor. The first three columns of Table 5 shows these results for three combinations of feature descriptor and success probability: 128-dimensional SIFT descriptors with success probabilities 0.9 and 0.95, and 512-dimensional GIST descriptors, with success probability 0.95.[2]

For each of these experiments, we estimate the exponent l for the weak separability condition. We assume that the number of candidates $\mathcal{T}$ retained has the form $\mathcal{T} = an^l$, and use l_1 regression and l_2 regression to estimate the parameters a and l. The bottom two rows of Table 5 plot the estimated a and l. In all cases considered, the estimated l is

[2]Notice that although the 512 dimensions Gist feature is not limited to $\mathbb{S}^{(d-1)}$, CP still maintains high success probabilities.

Size	SIFT-0.9 CP	SIFT-0.95 CP	SIFT-0.95 BHC [17]	Gist-0.95 CP
50K	52	118	1186	658
100K	64	147	2477	971
200K	91	208	3850	1754
500K	139	276	6543	2889
1M	181	389	9591	5201
l_1	$a = 0.498$ $l = 0.427$	$a = 1.586$ $l = 0.398$	$a = 0.223$ $l = 0.728$	$a = 2.887$ $l = 0.587$
l_2	$a = 0.472$ $l = 0.431$	$a = 1.435$ $l = 0.404$	$a = 0.249$ $l = 0.719$	$a = 2.364$ $l = 0.602$

Table 1. The number $\mathcal{T}$ to achieve the given accuracy. SIFT-0.9: SIFT feature with 0.9 accuracy. SIFT-0.95: SIFT feature with 0.95 accuracy. Gist-0.95: Gist feature with 0.95 accuracy. BHC is much less competitive, we omit the experiment on GIST.

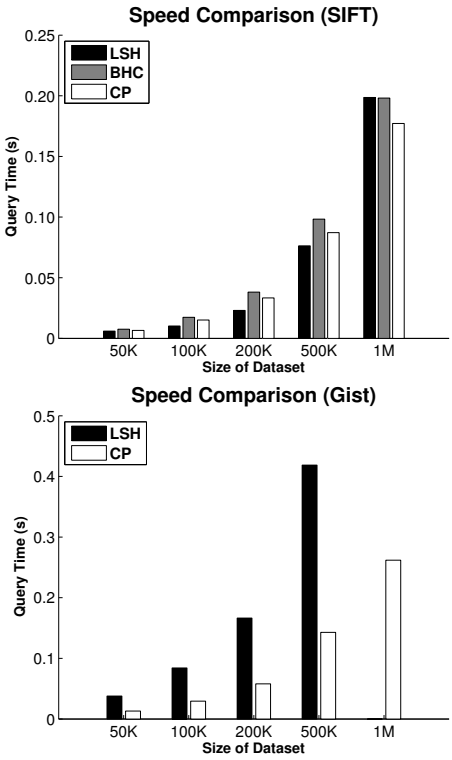

Figure 3. Speed comparison with LSH and BHC. The first figure shows the comparison with 128 dimensions SIFT feature. Whereas the latter one shows the comparison with 512 dimensions Gist features. Parameters of both algorithms are configured so that the accuracy of finding the exact NN is equal to 0.95. And the optimal parameters of LSH were chosen by the E2LSH.

much smaller than 1, indicating that that the second stage of the compact projection algorithm can be achieved in sublinear time. For SIFT descriptors, the distance of the query to only a minuscule fraction (less than 0.0002) of the database vectors needs to be computed to achieve a 90% probability of returning the exact nearest neighbor! Moreover, it reasonable to assume all the compact codes are stored in memory, since 2G memory can be used to store over 60M compact projections.

Table 5 also compares our approach to the Binary Hash Coding (BHC) algorithm proposed in [17]. That algorithm has a similar structure: hashing to select a subset of $\mathcal{T}$ candidates, followed by exact nearest neighbor search on the selected points. Comparing the second and third columns of Table 5, we see that for 95% success probability with SIFT descriptors, BHC requires that we retain at least 10 times as many candidates as CP. Intuitively, every bit generated by CP captures the global configuration of the vector. On the other hand, the bits generated by BHC captures only local information (based on means of each dimension), which is likely to be reluctant since some dimensions are highly correlated.

Comparison with Existing Approaches. We further compare the speed and memory requirements of CP to those of BHC [17] and LSH [2]. For this experiment, we use the *INRIA ANN* dataset of 1M SIFT features and our *Flickr* dataset of 1M GIST features. To fairly compare the quality of bits, the number of bits used by BHC is also set to 256. For LSH, we use the highly optimized implementation E2LSH. The experiment is done on a Pentium 4 1.6GHz machine with 4GB memory, using Debian Linux x64.

Figure 3 compares the average retrieval times for the three algorithms. Across the whole INRIA database, our approach slightly outperforms its two competitors in terms of speed. Moreover, as shown demonstrated in Figure 3 (bottom), CP significantly outperforms LSH across the Flickr database. In fact, for 1M GIST descriptors, E2LSH fails to estimate a set of parameters needed to achieve 0.95 accuracy. These significant practical gains in retrieval time come partly as a result of the lower space complexity of CP. Figure 4 compares the memory requirements of CP to those of LSH for varying database sizes, demonstrating a savings of at least an order of magnitude.

Consistent with our analysis, CP demonstrates a distinct advantage over LSH in high-dimensional spaces. This allows us to index higher-dimensional features such as GIST, leading to good retrieval performance even with limited memory. It is noteworthy that the memory requirement for LSH appears flat in Figure 4 for the automatic parameter selection in the E2LSH package: with larger databases, the package sacrifices accuracy to fit into the 2 GB of available memory. Figure 5 gives several representative examples of near neighbors returned by CP in the Flickr image set.

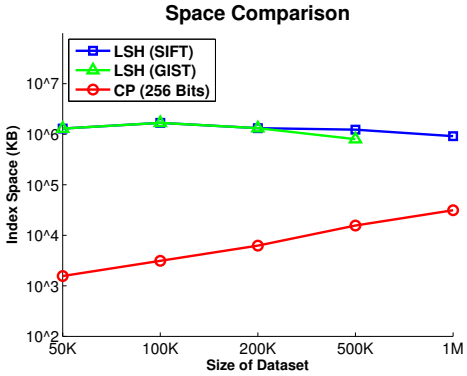

Figure 4. The index space comparison with LSH. The space complexity only depends on the number of projection bit and the size of the dataset.

6. Discussion

We have demonstrated that a very simple scheme, compact projection, gives a surprisingly efficient practical near neighbor search, with an explicit estimate of the required number of bits. The memory cost is several orders of magnitude smaller than alternatives such as LSH. This scheme can be applied in parallel and distributed computing environments with little modification. More work can be done on exploring the performance guarantee for unnormalized point sets. Given the increasing prevalence of high-dimensional data, we expect this scheme to have many applications, both in computer vision and beyond.

References

[1] D. Achlioptas. Database-friendly random projections. In *PODS*, 2001. 4

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006. 2, 4, 5, 6

[3] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, 2008. 4

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, 2004. 2, 4

[5] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, April 2008. 1

[6] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *SIGGRAPH*, 1980. 1

[7] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998. 1

[8] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42(6):1115–1145, 1995. 3

[9] W. Johnson and J. Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. 4

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2

[11] Y. Ohsawa and M. Sakauchi. Bd-tree: A new n-dimensional data structure with efficient dynamic characteristics. In *Proc. 9th World Computer Congress*. 1

[12] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001. 1, 2, 4

[13] J. T. Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *SIGMOD*, pages 10–18, 1981. 1

[14] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. 2006. 2, 4

[15] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, pages 1–8, June 2008. 1

[16] A. B. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 30(11):1958–1970, 2008. 2

[17] B. Wang, Z. W. Li, M. J. Li, and W. Y. Ma. Large-scale duplicate detection for web image search. *ICME*, 2006. 1, 5, 6

[18] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, 1998. 1

[19] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. 1

[20] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007. 1

[21] L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *ECCV Workshop on Faces in real-life images*, 2008. 1

Appendix: Proofs and Derivations

Proof of Lemma 2.

Proof. Let $\mathcal{B}(\boldsymbol{q}, r) := \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{q}\|_2 \leq r\}$ denote the closed ball with radius r centered at $\boldsymbol{q}$. For any $\boldsymbol{x}, \boldsymbol{y} \in B(\boldsymbol{q}, \Delta)$, $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \|\boldsymbol{x} - \boldsymbol{q}\|_2 + \|\boldsymbol{q} - \boldsymbol{y}\|_2 \leq 2\Delta$. So, if $B(\boldsymbol{q}, \Delta) \cap X$ is nonempty, then for any $\boldsymbol{x}_i \in B(\boldsymbol{q}, \Delta) \cap X$, $B(\boldsymbol{q}, \Delta) \cap X \subset B(\boldsymbol{x}_i, 2\Delta) \cap X$. Hence,

$$\max_j |\{i \mid \theta(\boldsymbol{x}_j, \boldsymbol{x}_i) \leq 2\Delta\}| \geq \max_{\boldsymbol{q} \in \mathbb{R}^d} |\{i \mid \theta(\boldsymbol{q}, \boldsymbol{x}_i) \leq \Delta\}|,$$

completing the proof. □

Proof of Theorem 1. Our proof of Theorem 1 relies on the following standard tail bound:

Lemma 3 (Chernoff Bound)**.** *Let $X = \sum_i X_i$ be a sum of independent random indicator variables X_i, and $\mu = E(X) = \sum_i E(X_i)$*

$$\forall \epsilon > 0, \qquad \mathbb{P}[X > (1+\epsilon)\mu] < e^{-\mu\epsilon^2/3};$$
$$\forall \epsilon \in (0,1), \qquad \mathbb{P}[X < (1-\epsilon)\mu] < e^{-\mu\epsilon^2/2}.$$

Figure 5. The top 8 nearest neighbors found in the 1M Flickr dataset with the 512 dimensions Gist feature. The query images are at the top left corners.

We now prove our main result:

Proof. Fix a query point $\boldsymbol{q} \in \mathbb{R}^d$. Let $\boldsymbol{x}_{(1)} \ldots \boldsymbol{x}_{(n)}$ denote the database points, sorted in order of increasing angle: $\theta(\boldsymbol{q}, \boldsymbol{x}_{(1)}) \leq \theta(\boldsymbol{q}, \boldsymbol{x}_{(2)}) \leq \ldots$. Let $\theta_1 \doteq \theta(\boldsymbol{q}, \boldsymbol{x}_{(1)})$, and $\theta_2 \doteq \theta(\boldsymbol{q}, \boldsymbol{x}_{(n^l)})$. We consider two cases.

Case 1: $\theta_2/\theta_1 \geq c$. We first bound the probability that $\|\boldsymbol{y}_{(1)} - \boldsymbol{y}_q\|_0 \geq k(\theta_1+\theta_2)/2$. Let $Z_i = |\boldsymbol{y}_{(1)}(i) - \boldsymbol{y}_q(i)|$. By Lemma 1, the Z_i are independent indicator random variables, with $\mathbb{E}[Z_i] = \theta_1/\pi$. Hence, $Z \doteq \|\boldsymbol{y}_{(1)} - \boldsymbol{y}_q\|_0 = \sum_{i=1}^n Z_i$, satisfies $\mathbb{E}[Z] = k\theta_1/\pi$, and

$$\mathbb{P}\left[Z > \frac{(\theta_1+\theta_2)k}{2\pi}\right] < \exp\left(-\frac{k\theta_1}{12\pi}\left(\theta_2/\theta_1 - 1\right)^2\right) \quad (2)$$

Since $\theta_2/\theta_1 \geq c$,

$$\begin{aligned}\theta_1(\theta_2/\theta_1 - 1)^2 &= (\theta_2 - \theta_1)(\theta_2/\theta_1 - 1) \\ &\geq (1-1/c)(c-1)\theta_2 > (1-1/c)(c-1)\Delta. \quad (3)\end{aligned}$$

Combining (3) and (2), setting $k \geq \frac{12\pi \ln(2/\delta)}{(1-1/c)(c-1)\Delta}$ gives

$$\begin{aligned}\mathbb{P}\left[X > \frac{(\theta_1+\theta_2)k}{2\pi}\right] &< \exp\left(-\frac{k}{12\pi}(1-1/c)(c-1)\Delta\right) \\ &\leq \delta/2 \quad (4)\end{aligned}$$

Let $F = \{n^l+1, n^l+2, \ldots n\}$. We next show it is very unlikely that n^l points $\boldsymbol{x}_{(j)}$ with $j \in F$ generate projections within distance $k(\theta_1+\theta_2)/2$ of $\boldsymbol{y}_q$. Fix $j \in F$, and now let $C_i = |\boldsymbol{y}_{(j)}(i) - \boldsymbol{y}_q(i)|$. Then $B_j \doteq \|\boldsymbol{y}_{(j)} - \boldsymbol{y}_q\|_0 = \sum_i C_i$ satisfies $\mathbb{E}(B_j) > k\theta_2/\pi$, and

$$\mathbb{P}\left[B_j < \frac{k(\theta_1+\theta_2)}{2\pi}\right] < \exp\left(-\frac{k\theta_2}{16\pi}\left(1-\theta_1/\theta_2\right)^2\right) \quad (5)$$

Now, let Q_j be the indicator random variable for the event that $\|\boldsymbol{y}_{(i)} - \boldsymbol{y}_q\|_0 < k(\theta_1+\theta_2)/2$, and let $Q \doteq \sum_{j\in F} Q_j$. Then

$$\begin{aligned}\mathbb{E}(Q) &< |F| \exp\left(-\frac{k\theta_2}{16\pi}\left(1-\theta_1/\theta_2\right)^2\right) \\ &\leq n \exp\left(-\frac{k\Delta}{16\pi}\left(1-1/c\right)^2\right). \quad (6)\end{aligned}$$

By Markov's inequality,

$$\mathbb{P}[Q > n^l] < \frac{\mathbb{E}(Q)}{n^l} < n^{(1-l)} \exp\left(-\frac{k\Delta}{16\pi}\left(1-1/c\right)^2\right)$$

Setting $k = \frac{16\pi(1-l)\ln n + \ln(2/\delta)}{(1-1/c)^2\Delta}$ is sufficient to ensure the probability is bounded by $\delta/2$. Combining the above two cases, if

$$\begin{aligned}k &= \max\left(\frac{12\pi\ln(2/\delta)}{(1-1/c)(c-1)\Delta}, \frac{16\pi(1-l)\ln n + \ln(2/\delta)}{(1-1/c)^2\Delta}\right) \\ &= O\left(\frac{\ln(2/\delta) + \ln n}{(1-1/c)^2\Delta}\right) \quad (7)\end{aligned}$$

then with probability at least $1-\delta$, $\boldsymbol{y}_{(1)}$ is one of the $2n^l$ closest codes to $\boldsymbol{y}_q$ in Hamming distance.

Case 2: $\theta_2/\theta_1 < c$**.** Notice that in this case, the $\mathcal{T}$-th near neighbor is a c-NN. Thus there are at least $\mathcal{T}$ possible solutions. It can be verified that the probability of not finding one of them can be easily bounded by δ.

Notice that in the above proof, the angle between two vectors is used to define the distance, the next lemma shows this is equivalent to the Euclidean distance up to a small constant factor.

Lemma 4. *For a point set lying on the unit sphere, the angle of two points gives good approximation of their Euclidean distance:* $d(\boldsymbol{p}, \boldsymbol{q}) \sim \theta(\boldsymbol{p}, \boldsymbol{q})$.

Proof. Note that

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\|\boldsymbol{p}\|_2^2 + \|\boldsymbol{q}\|_2^2 - 2\boldsymbol{p}\cdot\boldsymbol{q}} = \sqrt{2 - 2\cos\theta} \quad (8)$$

Using Taylor's expansion, we get

$$\sqrt{\theta^2 - \theta^4/12} \leq \sqrt{2-2\cos\theta} \leq \theta \quad (9)$$

It's easy to see that $\lim_{\theta\to 0} \frac{\theta}{\sqrt{2-2\cos\theta}} = 1$. Furthermore, since $\theta_a < \theta_b \iff \sqrt{2-2\cos\theta_a} < \sqrt{2-2\cos\theta_b}$, the estimation is order-preserving and tight with respect to a small constant factor.

Thus, we either find an exact nearest neighbor, or a c-approximate near neighbor by examining the Euclidean distances of at most $O(n^l)$ points.

□