Optimized Transform Coding for Approximate KNN Search

Minwoo Park
mpark@objectvideo.com
Kiran Gunda
kgunda@objectvideo.com
Himaanshu, Gupta
hgupta@objectvideo.com
Khurram, Shafique
kshafique@objectvideo.com

Research and Development Services
ObjectVideo
11600 Sunrise Valley Dr, Ste 210
Reston, USA
http://www.objectvideo.com

Abstract

Transform coding (TC) is an efficient and effective vector quantization approach where the resulting compact representation can be the basis for a more elaborate hierarchical framework for sub-linear approximate search. However, as compared to the state-of-the-art product quantization methods, there is a significant performance gap in terms of matching accuracy. One of the main shortcomings of TC is that the solution for bit allocation relies on an assumption that probability density of each component of the vector can be made identical after normalization. Motivated by this, we propose an optimized transform coding (OTC) such that bit allocation is optimized directly on the binned kernel estimator of each component of the vector. Experiments on public datasets show that our optimized transform coding approach achieves performance comparable to the state-of-the-art product quantization methods, while maintaining learning speed comparable to TC.

1 Introduction

Given a high dimensional query representation, retrieval of a few closest representations from a large scale (up to billions) high dimensional data set has been at the heart of many computer vision problems, such as image/video retrieval, image classification, object/ scene/ place recognition. Despite prolonged study, the problem of efficiently finding nearby points in high dimensions remains unsolved. This difficulty has led to the development of approximate nearest neighbor (ANN) search such as locality-sensitive hashing (LSH) [3], randomized KD-trees [19], hierarchical k-means [15], spectral hashing [21], product quantization [4, 8, 9, 16], and transform coding [2].

Code compactness is also important in the context of large-scale retrieval as memory size is often the primary determinant of system performance. For example, one billion data of 64-bit codes can still be loaded into memory to utilize memory DB. However, LSH [3] does not produce compact codes due to its randomized nature. In addition, there are many other alternative approaches such as randomized KD-trees [19] and hierarchical k-means [15] that outperform LSH.

In general, the quality of learned nodes using hierarchical k-means [15] and KD-trees [19] degrades as dimension of the input data grows and the sample data becomes sparse. More importantly, an erroneous decision made at the top level propagates during tree traversal.

Spectral hashing [21] achieves code efficiency which is a significant improvement over LSH 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. For example, a feature vector A can be mapped to a hash $H(A)$. In this case, of particular interest could be a search of K nearest neighborhood (KNN) of A in the original space. However, it is not easy to compute a list of KNN since there is a large difference between the neighborhoods of $H(A)$ and that of A.

Jegou et al. [8] propose a product quantization (PQ) method where input vectors are partitioned into a predetermined number of equal-sized sub-spaces. Then sub-vectors in each sub-space are quantized independently using k-means with a constant number of centers determined by bits per sub-space and Cartesian product of each of the independently estimated centers can produce codewords to represent the original space. This approach can generate millions of codewords efficiently for high dimensional data as compared to k-means on the original space.

Ge et al [4] propose optimized product quantization (OPQ) improving PQ by transforming input vectors such that each sub-space becomes less dependent on each other and the sum of eigenvalues of each sub-space is balanced. There is a significant improvement in terms of KNN search accuracy over the original PQ [8]. However, the use of k-means method for each sub-space can still be prohibitive when the size of the training set is large and the data is high dimensional, since the computation complexity of k-means is quadratic in the size of the training set, and linear in the data dimension.

Brandt [2] proposes a transform coding (TC) method. Brandt's method [2] is a very simple and efficient quantization technique using transform coding and a scalar PQ. Although it has good performance in terms of KNN search accuracy with greater speed, simplicity, and generality, OPQ [4] outperforms TC significantly.

Motivated by these, we propose optimized transform coding to improve the KNN search performance of TC. Our optimized transform coding (OTC) estimates underlying probability density of the PCA coefficients by binned kernel estimator, and then performs approximate Lloyd-Max algorithm [11, 13] on the estimated probability density. After that, we use a novel reformulation of the bit allocation problem to make it computationally tractable. Our proposed OTC approach has speed, simplicity, and generality similar to TC [2], with KNN accuracy comparable to the state of the art OPQ [4].

2 Optimized Transform Coding

We introduce our optimized transform coding (OTC) in the context of general vector quantization. A quantizer encoder $e(\mathbf{x})$ is a real-valued function $\mathcal{E} : \mathbb{R}^n \to \mathcal{I}$ characterized by the region it induces on the input space, $\mathcal{R}^n_{\mathbf{x}} = \{\mathbf{x} \in R^n(i) : e(\mathbf{x}) = i\}$ and $\cup_{i=1}^{L} R^n(i) = \mathbb{R}^n$ where $\mathcal{I} = \{1, \cdots, L\}$ and $\mathbf{x}$ is an input vector. The decoder $d(i)$ is a real-valued function $\mathcal{D} : \mathcal{I} \to \mathbb{R}^n$ characterized by the codebook $\mathcal{C} = \{i \in \mathcal{I} : d(i) = y_i\} \subset \mathbb{R}^n$. The mean distortion error of the given quantization level L (MDE) of the quantization is given as:

$$MDE(L) = \sum_{i=1}^{L} \int_{R^n(i)} f(\mathbf{x}) Dist(\mathbf{x}, d(e(\mathbf{x}))) d\mathbf{x} \tag{1}$$

where f is an estimated probability density function of multi-dimensional vector $\mathbf{x}$ and $Dist(\mathbf{x}, \mathbf{x}')$ is a distortion error between $\mathbf{x}$ and $\mathbf{x}'$.

In general, to find the optimal set of region $\mathcal{R}_\mathbf{x}$, the codebook $\mathcal{C}$, and the given quantization level L, minimum-distortion quantizer aims to minimize mean distortion error (MDE) as follows:

$$\left(\mathcal{R}_\mathbf{x}^{opt}, \mathcal{C}^{opt}\right) = arg \min_{\mathcal{R}_\mathbf{x}, \mathcal{C}} MDE(L) \tag{2}$$

Although design of such a scalar quantizer to satisfy the minimum distortion criterion is well understood, vector quantization is still an open problem. For instance, it can be challenging to obtain sufficient sample data to characterize $f(\mathbf{x})$. Moreover, solving Eq. (2) is computationally expensive in high dimensions.

However, if $p(\mathbf{x})$ is independent in its components (dimensions), and the metric is of the form given as:

$$Dist(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{D} dist(x_k, x'_k), \tag{3}$$

where D is a dimension of $\mathbf{x}$, $\mathbf{x}_k$ are the k^{th} component of $\mathbf{x}$, and $dist(x_k, x'_k)$ is a distance metric between $\mathbf{x}_k$ and $\mathbf{x}'_k$, we can obtain a minimum distortion quantizer by forming the Cartesian product of the independently quantized components. That is, the vector quantization encoder can be of a form, $e(\mathbf{x}) = [e_1(x_1), \cdots, e_D(x_D)]^T$. In the original PQ [8, 9], D dimensional space is divided into M sub-spaces (typical M is 8) to form given as:

$$e(\mathbf{x}) = [e_{1\sim K}(x_{1\sim K}), \cdots, e_{7K+1\sim 8K}(x_{7K+1\sim 8K})]^T \text{ where } K = D/M. \tag{4}$$

However, each component is not independent in practice. Therefore, TC [2] and OPQ [4] aim to minimize inter-component dependencies using the principal component analysis (PCA) and show great success over the original PQ [8, 9]. After minimizing the inter-component statistical dependencies using PCA, the quantizer design problem reduces to a set of M number of independent K dimensional problems. In TC, $K = 1$ and $M = D$. The major difference between OPQ and TC lies in the bit-allocation approach used in each method. The key difference is that OPQ assigns the same number of bits per sub-space, while TC assigns a different number of bits per sub-space. Therefore OPQ finds the best combination of components for each sub-space while maintaining the same number of bits for each sub-space while TC finds the number of bits suitable for each sub-space.

In the context of TC, each quantizing encoder e_k at the k^{th} dimension is designed independently for every $1 \le k \le D$ to minimize the expected distortion given as:

$$MDE_k(L_k) = \sum_{i=1}^{L_k} \int_{R_k(i)} f_k(c_k) dist_k(c_k, d_k(e_k(c_k))) dc_k. \tag{5}$$

where c_k is PCA coefficient after projection of $\mathbf{x}$ to PCA subspace k.

Therefore, a vector quantization using B-bits code is summarized as follows:

$$(\mathcal{L}, \mathcal{R}_\mathbf{c}, \mathcal{C})^{opt} = arg \min_{\mathcal{L}, \mathcal{R}_\mathbf{c}, \mathcal{C}} \sum_{k=1}^{D} MDE_k(L_k) \text{ subject to } \sum_{k=1}^{D} log_2(L_k) = B. \tag{6}$$

If the number of distinct quantization levels per k^{th} component L_k is known for a total target bit B, a product quantizer can be obtained by using the minimum distortion criterion.

Optimal bit allocation is achieved by minimizing the expected distortion due to quantization. However, solution to this optimization problem for general distributions and distortion functions requires computationally prohibitive numerical search [2].

Instead, Brandt [2] adopted greedy integer-constrained allocation algorithm [5] to assign bits. Number of the quantization level set to be proportional to the variance of the data under the two assumptions that 1) probability density of each component can be made identical after the normalization and 2) per-component distortion functions are identical. However, the first assumption can be easily violated in many cases (e.g., non-Gaussian probability density function). Motivated by this problem, we propose to solve Eq. (6) directly in our proposed optimized transform coding (OTC).

2.1 Problem Statement

In our proposed OTC, the optimal bit allocation problem is formulated as a constrained minimization problem as:

$$(\mathcal{L},\mathcal{R}_{\mathbf{c}},\mathcal{C},T)^{opt} = arg \min_{\mathcal{L},\mathcal{R}_{\mathbf{c}},\mathcal{C},T} \left(Obj(\mathcal{L},\mathcal{R}_{\mathbf{c}},\mathcal{C},T) = \left[\sum_{k=1}^{T} MDE_k(L_k)\lambda_k + \sum_{k=T+1}^{D} E_k\lambda_k \right] \right) \tag{7}$$

subject to $\sum_{k=1}^{T} log_2(L_k) = B$ where $Obj(\mathcal{L},\mathcal{R}_{\mathbf{c}},\mathcal{C},T)$ is a nonlinear mean distortion error of the quantization, λ_k is an eigenvalue of k^{th} subspace of PCA, E_k is an information loss due to dimensionality reduction of PCA, T is the dimension after dimensionality reduction, B is the number of target bits, and D is the dimension of the original vector.

In solving Eq. (7), there are three major challenges to overcome.

1. How can we estimate $f_k(c_k)$ efficiently for millions of c_k for $1 \le k \le D$?
2. How can we optimally estimate $\mathcal{L},\mathcal{R}_{\mathbf{c}}$ efficiently for millions of c_k for $1 \le k \le D$?
3. How can we minimize $Obj(\mathcal{L},\mathcal{R}_{\mathbf{c}},\mathcal{C},T)$, a nonlinear system subjects to integrality requirements for the variables?

In this paper, we address each of the above challenges in the following sections.

2.2 Efficient Estimation of Density Function of PCA coefficients

Accurate and efficient estimation of probability density functions of PCA coefficients $f_k(c_k)$ in Eq. (5) is an important step to optimize Eq. (7). We first perform Principal Component Analysis (PCA) to minimize inter-component dependencies. Then we perform binned kernel estimation (BKE).

Principal Component Analysis: For a set of N number of $D \times 1$ vectors given as $\mathbf{x_1}, \cdots, \mathbf{x_N}$, a principal component analysis is performed for a covariance matrix given as:

$$\mathbf{Cov} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i}-\mathbf{x_m})(\mathbf{x_i}-\mathbf{x_m})^{\mathbf{T}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i^T} - \mathbf{x_m^2} = \mathbf{P\Sigma P^T} \tag{8}$$

where $\mathbf{P} = [\mathbf{p_1}, \cdots, \mathbf{p_D}]$ is projectction matrix of eigenvectors, Σ=$diag(\lambda_1, \cdots, \lambda_D)$ is a diagonal matrix with eigenvalues in descending order, and $\mathbf{x_m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}$.

In this manner, we can perform PCA efficiently for very large N since the memory requirement of Eq. (8) does not depend on N but on $D \times D$ and the computation of Eq. (8) can be easily parallelizable where D is the dimension of $\mathbf{x}_i$. The PCA coefficients are given as $\mathbf{c_i} = \mathbf{P^T}(\mathbf{x_i} - \mathbf{x_m})$. For these coefficients, we perform weighted density estimation for each dimension $d = 1 \sim D$ using a weighted Parzen Window estimation below.

Binned Kernel Estimator (BKE): The use of BKE enables to deal with millions of floating data efficiently since its memory requirement does not depend on the number of training data N but on the number of bins. For all k^{th} PCA coefficients c_k, we compute BKE. For $D \times N$ PCA coefficients matrix[1]:

$$\mathbf{C} = [\mathbf{c}(1), \cdots, \mathbf{c}(N)] \text{ where } \mathbf{c}(j) = [c_1(j), \cdots c_D(j)]^T, \tag{9}$$

we first compute D number of normalized histogram for $\mathbf{c}_k = [c_k(1), \cdots, c_k(N)]$ where $\mathbf{c}_k$ is a k^{th} row vector of $\mathbf{C}$ where $1 \le k \le D$. The computed histogram for $\mathbf{c}_k$ is given as a set of bin locations $\dot{s}_k^{[i]}$ and their weights $w_k^{[i]}$ where i is an index of bin. The bin size is given as $\dot{s}_k^{[i+1]} - \dot{s}_k^{[i]} = \alpha \times \varepsilon_{MF}$ where ε_{MF} is the machine epsilon of float type. Therefore, the total number of bins is given as $M(k) = \frac{max\left(\dot{s}_k^{[i]}\right) - min\left(\dot{s}_k^{[i]}\right)}{\alpha\varepsilon_{MF}}$. In this paper, we set $\alpha = 50$. In this manner, the maximum number of bins is fixed regardless of N (e.g., there are about 166,667 number of bins for c_k with $0 \le c_k \le 1$ range in C++ since $\varepsilon_{MF} = 1.19209 \times 10^{-7}$).

The created histogram for dimension k is given as $H(k) = \left\{ \left(\dot{s}_k^{[i]}, w_k^{[i]}\right) | 1 \le i \le M(k) \right\}$. For the set of $\left(\dot{s}_k^{[i]}, w_k^{[i]}\right)$, the weighted density estimation is given as:

$$f_k(s_k) = \sum_{i=1}^{\mathcal{R}_k} K_h(s_k - \dot{s}_k^{[i]}) w_k^{[i]}. \tag{10}$$

Note that the evaluation of Eq. (10) requires constant time since $\dot{s}_k^{[i]}$ is ordered scalar value and $K_h(s_k - \dot{s}_k^{[i]})$ produces zero values outside of the kernel bandwidth. Therefore, the evaluation of $f_k(s)$ for all $\dot{s}_k$ takes $O(N)$. We precompute $f_k(\dot{s}_k^{[i]})$ for all i and k to construct 2 dimensional table $T_f(k,i)$ where $T_f(k,i)$ holds $f_k(\dot{s}_k^{[i]})$, $1 \le k \le D$, and $1 \le i \le M(k)$.

2.3 Approximate Lloyd-Max Algorithm on the BKE

With the binned kernel estimator of PCA coefficient c_k with $1 \le k \le D$, we can use the results of Lloyd-Max algorithm. The mean distortion of the quantization (MDE) is measured by the mean squared error (MSE) given as:

$$MDE_k(L_k) = MSE_k(L_k) = \sum_{i=1}^{L_k} \int_{R_k(i)} f_k(s_k) \left[s_k - d\left(e\left(s_k\right)\right)\right]^2 ds_k. \tag{11}$$

where the subscript k stands for dimension index of the PCA coefficients in the previous section.

In general, to find the optimal set of boundaries $\mathcal{R}_k$, the codebook $\mathcal{C}_k$, and the given quantization level L_k, Llyod-Max algorithm [11, 13] aims to minimize MSE as follows:

$$\left(\mathcal{R}_k^{opt}, \mathcal{C}_k^{opt}\right) = argmin_{\mathcal{R}_k, \mathcal{C}_k} MSE_k(L_k) \tag{12}$$

We repeat the steps of Lloyd-Max algorithm:

$$\text{Step 1: } y_i = \frac{\int_{t_i}^{t_{i+1}} s_k f_k(s_k) ds_k}{\int_{t_i}^{t_{i+1}} f_k(s_k) ds_k}, \qquad \text{Step 2: } t_i = \frac{y_{i+1} + y_i}{2} \tag{13}$$

until convergence where $R_k(i) = \{s_k | t_{i-1} \le s_k < t_i\}$.

[1]The $D \times N$ PCA coefficient matrix is shown for explanation purpose only.

However, evaluating $\int_{t_i}^{t_{i+1}} s_k f_k(s_k) ds_k$ and $\int_{t_i}^{t_{i+1}} f_k(s_k) ds_k$ for every iteration using a standard numerical integration is prohibitive since optimal set of L_k for $1 \le k \le T$ and T are not known. Therefore the integration in "Step 1" is replaced by our proposed integral approximation as follows:

$$\int_{s_k^{[n]}}^{s_k^{[m]}} f_k(s_k) s_k ds_k \approx \frac{\Delta s_k}{\alpha \varepsilon_{MF}} \left[f_k(\dot{s}_k^{[m+1]}) \dot{s}_k^{[m+1]} - f_k(\dot{s}_k^{[n]}) \dot{s}_k^{[n]} \right] + \left(\frac{\Delta s_k^2}{\alpha \varepsilon_{MF}} - \Delta s_k\right) \left[f_k(\dot{s}_k^{[m+1]}) - f_k(\dot{s}_k^{[n]}) \right] + \sum_{i=0}^{m} f_k(\dot{s}_k^{[i]}) \dot{s}_k^{[i]} - \sum_{i=0}^{n-1} f_k(\dot{s}_k^{[i]}) \dot{s}_k^{[i]} \tag{14}$$

$$\int_{s_k[n]}^{s_k[m]} f_k(s) ds \approx \frac{\Delta s_k}{\alpha \varepsilon_{MF}} \left[f_k(\dot{s}_k^{[m+1]}) - f_k(\dot{s}_k^{[n]}) \right] + \sum_{i=0}^{m} f_k(\dot{s}_k^{[i]}) - \sum_{i=0}^{n-1} f_k(\dot{s}_k^{[i]}) \tag{15}$$

The derivation of Eqs. (14) and (15) can be found in an accompanied supplementary material.

The evaluation of $\sum_{i=0}^{m} f_k(\dot{s}_k^{[i]})$ and $\sum_{i=0}^{m} f_k(\dot{s}_k^{[i]}) \dot{s}_k^{[i]}$ in Eqs. (14) and (15) take $O(N)$ at most and they can be precomputed to enable constant time evaluation of Eqs. (14) and (15). The value of $\sum_{i=0}^{m} f_k(\dot{s}_k^{[i]})$ is stored in table $T_{cdf}(k,m)$ and the value of $\sum_{i=0}^{m} f_k(\dot{s}_k^{[i]}) \dot{s}_k^{[i]}$ is stored in table $T_{cdfx}(k,m)$. Eq. (13) repeats until convergence using the precomputed tables $T_{cdf}(k,m)$, $T_{cdfx}(k,m)$, and $T_f(k,m)$(Section 2.2).

Since Eq. (13) can be computed in constant time and is independent of N, this process finishes in constant time as well[2]. Fig. 1 shows a distribution of the first PCA coefficient in one of our evaluation datasets (SIFT1M [9]) and its quantization using our proposed methods. It shows that the distribution in this case is bimodal.

2.4 Non-linear Constrained System subjects to Integrality

Eq. (7) is a nonlinear system subject to integrality requirements for the variables. Research efforts in the past fifty years have led to development of linear integer programming as a mature discipline of mathematical optimization. Such a level of maturity has not been reached when one considers nonlinear systems subject to integrality requirements for the variables. Although, there are several approaches such as simulated annealing and genetic algorithm to solve this problem, such solutions generally require heavy and lengthy computation and cannot guarantee the integral solution.

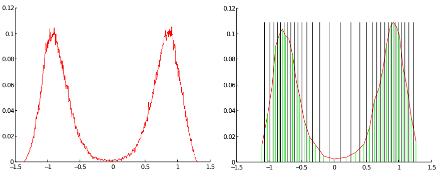


Figure 1: Left: Binned Kernel Estimation for c_1 of sift1M data set. Right: Results of Lloyd-Max Algorithm. The green vertical lines are y_i and the black vertical lines are t_i in Eq. (13) where L_1 is 32 (5 bits).

The approach introduced by Shoham and Gersho [18] could be used to compute the optimal bit allocation where they solve the bit allocation problems using the Lagrange multiplier method and dynamic programming. However, it requires good initialization and the number of necessary iterations can be significant. Other variants of the algorithm have been developed to overcome the requirement of good initialization with convex assumption of rate-distortion function. In any case, a closed form of distortion function is required to find the optimal bit allocations and that prevents us from using these algorithms.

[2]The algorithm converges after $10 \sim 20$ number of iterations.

Algorithm 1 Optimized Transform Coding, $Q = OTC(\mathbf{x_i}, B, N, \mathbf{K_h}, \alpha)$

```
 1: Compute $\mathbf{Cov} = \mathbf{P\Sigma P^T} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i x_i^T} - \left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}\right]^2$
 2: Compute $\mathbf{c_i}$ for all $i$ by $\mathbf{c_i} = \mathbf{P^T}(\mathbf{x_i} - \mathbf{x_m})$
 3: Compute histogram $\mathbf{H}(\mathbf{k})$ for all $k$, $1 \le k \le D$ using $\alpha$
 4: Compute $T_f(k,i) = f_k(\dot{s}_k^{[i]})$ for all $k$ and $i$ using a kernal $\mathbf{K_h}$
 5: Compute $T_{cdf}(k,i) = \sum_{j=1}^{i} f_k(\dot{s}_k^{[j]})$ for all $k$ and $i$
 6: Compute $T_{cdfx}(k,i) = \sum_{j=1}^{i} f_k(\dot{s}_k^{[j]})\dot{s}_k^{[i]}$ for all $k$ and $i$
 7: Initialize 2 dimensional sparse arrays, $\mathbf{A}_{MDE}(\cdot,\cdot), \mathbf{A}_{\mathcal{R}}(\cdot,\cdot), \mathbf{A}_{\mathcal{C}}(\cdot,\cdot)$
 8: Find a feasible solution set $\mathbf{S} = \{S_i | 1 \le i \le N_s\}$ by Diophantine solver [1]
 9: for all  $S_i$ for $1 \le i \le N_S$ do
10:     $Obj = 0$
11:     for all $L_k$ for $1 \le k \le D$ do
12:         if $k \le T(i)$ then
13:             if $\mathbf{A}_{MDE}(k,L_k) == \emptyset$ then
14:                 $\left(\mathcal{R}_k^{opt}, \mathcal{C}_k^{opt}, MDE^{opt}\right) = argmin_{\mathcal{R}_k,\mathcal{C}_k,MDE_k} MDE_k(L_k)$
15:                 $\mathbf{A}_{MDE}(k,L_k) = MDE^{opt}$, $\mathbf{A}_{\mathcal{R}}(k,L_k) = \mathcal{R}_k^{opt}$, $\mathbf{A}_{\mathcal{C}}(k,L_k) = \mathcal{C}_k^{opt}$
16:             end if
17:             $Obj+ = \mathbf{A}_{MDE}(k,L_k)$, $\mathcal{R}_k = \mathbf{A}_{\mathcal{R}}(k,L_k)$, $\mathcal{C}_k = \mathbf{A}_{\mathcal{C}}(k,L_k)$
18:         else
19:             $Obj+ = E_k$
20:         end if
21:     end for
22:     $E_{MSE}(i) \leftarrow Obj$
23:     $Q(i) \leftarrow \mathcal{R} = \{\mathcal{R}_1,\cdots,\mathcal{R}_{T(i)}\}$, $\mathcal{C} = \{\mathcal{C}_1,\cdots,\mathcal{C}_{T(i)}\}$
24: end for
25: $i_{min} = argmin_i E_{MSE}(i)$
26: return $Q = Q(i_{min})$
```

However, we propose the following constraints to assign more number of bits to the PCA dimension having larger variance:

$$\sum_{k=1}^{T} log_2(L_k) = B \text{ where } L_1 \ge L_2 \ge \cdots \ge L_T,\ log_2(L_1) \le B_c, \text{ and } B_c \le B. \tag{16}$$

We then propose to formulate Eq. (16) as a system of linear Diophantine Eqs. [14] as follows:

$$\sum_{k=1}^{B_c} k \times n_k = B,\ \sum_{k=1}^{B_c} n_k = T,\ 0 \le n_k \le T \tag{17}$$

where n_k is the number of dimensions which are assigned k-bits. That is, Eq. (17) enforces that a weighted sum of product of monotonically increasing bits k and the number of dimensions n_k that use k bits should be B, and the sum of all dimensions n_k should be T. For example, for $B_c = 4$, $B = 12$, and $T = 5$, one of feasible solutions $(n_4,n_3,n_2,n_1) = (1,1,2,1)$ means $(L_1,L_2,L_3,L_4,L_5) = (2^4,2^3,2^2,2^2,2^1)$. We use a linear Diophantine solver introduced by Aardal et al. [1] to solve the general Diophantine Eqs. in Eq. (17). The results of the Diophantine solver can be precomputed and reused since the Diophantine solution is independent of the dataset.

The size of the solution space of Eq. (16) becomes numerically tractable with our proposed efficient Lloyd-Max algorithm on the binned kernel estimator. For all feasible solution set given as:

$$S_i=\{\underbrace{2^{B_c}\ldots2^{B_c}}_{n_{B_c}},\ \underbrace{2^{B_c-1}\ldots2^{B_c-1}}_{n_{B_c-1}},\ldots,\ \underbrace{2^2\ldots2^2}_{n_2},\ \underbrace{2^1\ldots2^1}_{n_1}\}=\{L_1,\ldots,L_{T(i)}\} \tag{18}$$

for $1 \le i \le N_S$ where $T(i)$ is the maximum dimension for the i^{th} solution and N_S is the total number of feasible solutions, we minimize $MSE_k(L_m)$ for all $1 \le m \le T(i)$ and for all $1 \le k \le D$ to compute the $Obj(\mathcal{L},\mathcal{R}_\mathbf{c},\mathcal{C},T)$ in Eq. (7). To enable efficient computation, the computed $MSE_k(L_m)$ is stored in a sparse array. Algorithm 1 summarizes our proposed OTC.

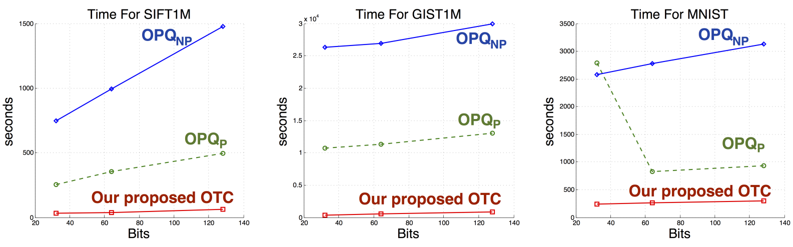
Steps 1 through 6 correspond to Section 2.2 and steps 7 through 26 correspond to Section 2.4 where step 14 corresponds to Section 2.3. For each PCA coefficient (c_k) of a vector, the codeword is assigned by the learned $R_k(i)$ as follows: c_k is assigned to codeword index i when $c_k \in R_k(i)$.

3 Experiments and Results

We evaluate the performance of our proposed optimized transform coding for three public dataset. The first two data sets are SIFT1M and GIST1M introduced by Jegou et al. [9] where SIFT1M contains 1 million 128 dimensional SIFT feature [12] vectors with 10K queries and GIST1M contains 1 million 960 dimensional GIST feature [17] vectors with 1K queries. The third data set is MNIST [10] which contains 70K images of hand written digits where each image has width and height of 28 $\times$ 28. We use the same 10k queries obtained from the authors of [4]. Following the experiment setting of [4], the top 100 nearest neighbors are considered as the true neighbors. The distance between a query and any vector is approximated by the distance of their codewords (known as Symmetric Distance Computation or SDC) and the data is sorted with respect to the rank. We compare the following methods:

OTC	Our optimized transform coding	**TC**	Transform coding [2]
OPQ$_{\mathbf{P}}$	Optimized product quantization with a parametric solution [4]	**OPQ$_{\mathbf{NP}}$**	Optimized product quantization with a non-parametric solution [4]
PQ$_{\mathbf{RO}}$	Product quantization with a random order [9]	**PQ$_{\mathbf{RR}}$**	Product quantization with PCA and random rotation [7]
ITQ	One of the state-of-the-art hashing method, a special vector quantization [6]		

Performance on Speed: On an average across multiple bit lengths, the speed of training using the proposed OTC is 8.15 times faster than **OPQ$_{\mathbf{P}}$** and 23.61 times faster than **OPQ$_{\mathbf{NP}}$** on SIFT1M [9]. On GIST1M [9], the OTC is 22.65 times faster than **OPQ$_{\mathbf{P}}$** and 54.32 times faster than **OPQ$_{\mathbf{NP}}$**. On MNIST [10], the OTC is 5.99 times faster than **OPQ$_{\mathbf{P}}$** and 10.61 times faster than **OPQ$_{\mathbf{NP}}$**. The detailed performance speed-up numbers for 32/64/128 bits are shown in Table 2 and Fig. 2. In our experiments, we do not use any non-exhaustive methods like inverted files [20], as that is not the focus of this paper. Detailed computation times for each of the steps in our proposed OTC pipeline can be found in the Table 1.



(a) SIFT1M [9] (b) GIST1M [9] (c) MNIST [10]

Figure 2: Timing measured in seconds on a Intel i7 2.6Ghz machine. (a) Size of learning set is 100,000 with 128 dimensional data. (b) Size of learning set is 500,000 with 960 dimensional data. (c) Size of learning set is 70,000 with 784 dimensional data. Note: For 32-bits, OPQ_P took longer than OPQ_{NP} since the number of dimension assigned to each subspace is quite large ($784/4 = 196$) and OPQ_P did not converge until max number of iterations is reached.

Table 1: Actual time in seconds for the optimized transform coding

Data	SIFT1M [9]			GIST1M [9]			MNIST [10]		
Bits	32	64	128	32	64	128	32	64	128
PCA (seconds)	1.2			125.1			39.9		
BKE (seconds)	25.1			103.9			64.4		
Optimization (seconds)	7.2	14.0	36.1	116.2	279.3	658.6	30.5	52.7	94.7
Query (seconds)	0.05	0.06	0.09	0.04	0.05	0.09	0.02	0.02	0.02

All of **OPQ$_{\mathbf{NP}}$**, **OPQ$_{\mathbf{P}}$**, and **OTC** have a computation time linear in the data dimension in order to learn optimal quantization. However, our proposed **OTC** has computation time linear in the size of the training data, whereas **OPQ$_{\mathbf{NP}}$** and **OPQ$_{\mathbf{P}}$** have a quadratic computation time. In addition, the quantization of input data using **OTC** has a $O(logK_i)$ computation time where K_i is the number of quantization levels at i^{th} dimension. This is so because OTC is a scalar product quantizer and an assignment of the i^{th} dimensional value of the input data to one of the quantization level can be done by using the range tree. However, the quantization using the **OPQ$_{\mathbf{NP}}$** or **OPQ$_{\mathbf{P}}$** has a $O(\hat{K}_i)$ computation time where $\hat{K}_i$ is the number of the quantization level at i^{th} subspace. Therefore, retrieval speed of OTC can be made much faster than **OPQ$_{\mathbf{NP}}$** and **OPQ$_{\mathbf{P}}$**. Note that the retrieval speed of OTC in the Table 1 is recorded using a brute-force quantization rather than using the desired range tree.

Table 2: Mean Average Precision (mAP) and Speed Improvement

MNIST [10]		**PQ$_{\mathbf{RR}}$**	**PQ$_{\mathbf{RO}}$**	**ITQ**	**TC**	**OPQ$_{\mathbf{P}}$**	**OPQ$_{\mathbf{NP}}$**	**OTC**	$\frac{\mathbf{OPQ_P}(sec)}{\mathbf{OTC}(sec)}$	$\frac{\mathbf{OPQ_{NP}}(sec)}{\mathbf{OTC}(sec)}$
32bit	mAP	0.26	0.31	0.30	0.30	**0.36**	**0.46**	**0.36**	11.72	10.83
64bit	mAP	0.35	0.45	0.50	0.51	**0.60**	**0.69**	**0.61**	3.16	10.65
128bit	mAP	0.47	0.64	0.67	0.68	**0.80**	**0.81**	**0.81**	3.08	10.36
GIST [9]		**PQ$_{\mathbf{RR}}$**	**PQ$_{\mathbf{RO}}$**	**ITQ**	**TC**	**OPQ$_{\mathbf{P}}$**	**OPQ$_{\mathbf{NP}}$**	**OTC**	$\frac{\mathbf{OPQ_P}(sec)}{\mathbf{OTC}(sec)}$	$\frac{\mathbf{OPQ_{NP}}(sec)}{\mathbf{OTC}(sec)}$
32bit	mAP	0.03	0.02	0.03	0.03	**0.05**	**0.06**	**0.05**	31.06	76.18
64bit	mAP	0.04	0.04	0.04	0.07	**0.14**	**0.14**	**0.13**	22.21	53.04
128bit	mAP	0.07	0.07	0.06	0.11	**0.30**	**0.30**	**0.29**	14.67	33.75
SIFT [9]		**PQ$_{\mathbf{RR}}$**	**PQ$_{\mathbf{RO}}$**	**ITQ**	**TC**	**OPQ$_{\mathbf{P}}$**	**OPQ$_{\mathbf{NP}}$**	**OTC**	$\frac{\mathbf{OPQ_P}(sec)}{\mathbf{OTC}(sec)}$	$\frac{\mathbf{OPQ_{NP}}(sec)}{\mathbf{OTC}(sec)}$
32bit	mAP	0.06	**0.08**	0.04	**0.08**	**0.09**	**0.11**	**0.08**	7.70	22.37
64bit	mAP	0.12	0.20	0.11	0.19	**0.24**	**0.26**	**0.25**	8.82	24.75
128bit	mAP	0.28	0.47	0.22	0.37	**0.54**	**0.55**	**0.56**	7.93	23.70

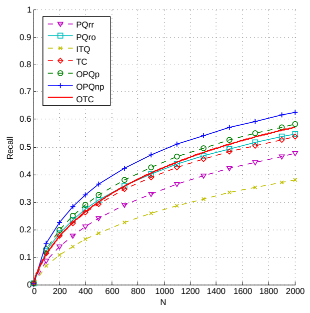
Performance on Accuracy: We compare all results in terms of mean average precision (mAP) and recall *vs.* N. As can be seen in Table 2 and Figure 3, our proposed **OTC** outperforms **PQ$_{\mathbf{RO}}$**,**PQ$_{\mathbf{RR}}$**,**TC**, and **ITQ** for all dataset in terms of mean average precision (mAP) and recall *vs.* N. Bold characters indicate top 3 performers, the red font color indicates the top performer among all methods, and the blue font color shows baseline **TC** method to emphasize **OTC**'s improvement.

In general, as the number of bits increases, our proposed **OTC** performs on par with **OPQ$_{\mathbf{P}}$** and **OPQ$_{\mathbf{NP}}$** in terms of accuracy, with almost 10+ fold speed improvement (See Table 2 and Fig. 2). All of the reported results, except ours, are provided by Ge et al. [4].

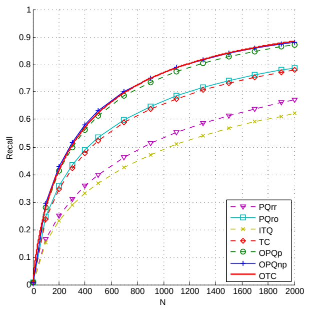
4 Conclusion

We have proposed optimized transform coding to improve the KNN search performance of TC. Our optimized transform coding estimates underlying probability density of the PCA coefficients by binned kernel estimator, and then performs approximate Lloyd-Max algorithm [11, 13] on the estimated probability density. After that, we use a novel reformulation of the bit allocation problem to make it computationally tractable. Our proposed OTC approach has speed, simplicity, and generality similar to TC [2], with KNN accuracy comparable to the state of the art OPQ [4].

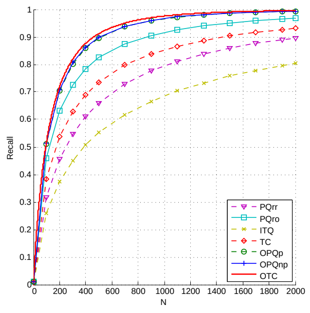
Recall *vs.* N for SIFT1M [9]



(a) 32 bits

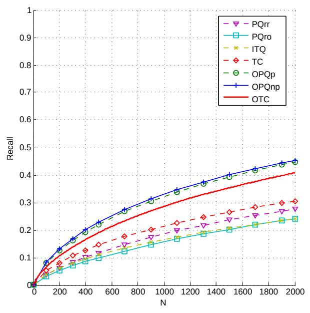


(b) 64 bits

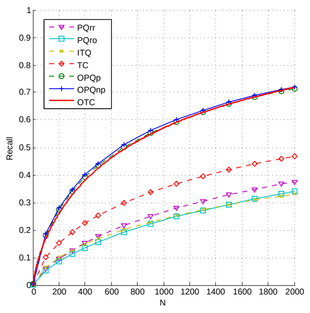


(c) 128 bits

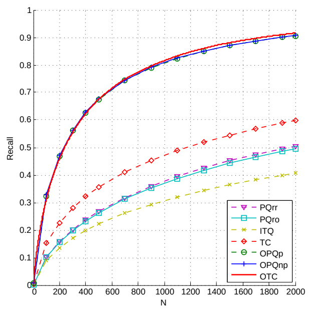
Recall *vs.* N for GIST1M [9]



(d) 32 bits

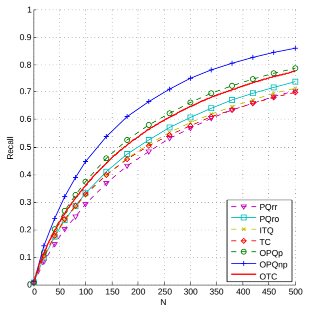


(e) 64 bits

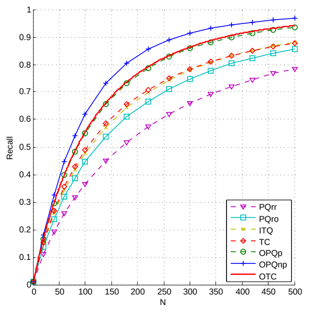


(f) 128 bits

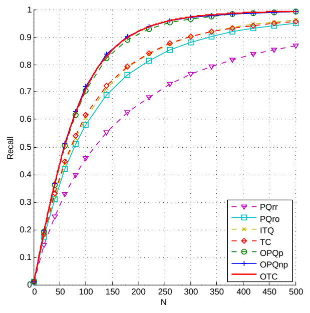
Recall *vs.* N for MNIST [10]



(g) 32 bits



(h) 64 bits



(i) 128 bits

Figure 3: Top: Recall curve for SIFT1M [9] data set is shown for 32, 64 and 128 bits. Our proposed OTC outperforms all other methods for 128 bit coding and has comparable performance to the state-of-the-art OPQ method. Middle: Recall curve for GIST1M [9] data set is shown for 32, 64 and 128 bits. Our proposed OTC outperforms all other methods for 128 bit coding and has comparable performance to the state-of-the-art OPQ method. Bottom: Recall curve for MNIST [10] data set is shown for 32, 64 and 128 bits. Our proposed OTC has comparable performance to the state-of-the-art OPQ method.

Acknowledgments: Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C-7212. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

[1] Karen Aardal, Cor A. J. Hurkens, and Arjen K. Lenstra. Solving a system of linear diophantine equations with lower and upper bounds on the variables. *Math. Oper. Res.*, 25(3):427–442, 2000.

[2] Jonathan Brandt. Transform coding for fast approximate nearest neighbor search in high dimensions. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 1815–1822, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[3] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[4] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 2946–2953, Washington, DC, USA, 2013. IEEE Computer Society.

[5] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991. ISBN 0-7923-9181-0.

[6] Yunchao Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 817–824, Washington, DC, USA, 2011. IEEE Computer Society.

[7] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 3304–3311, June 2010.

[8] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the* 10^{th} *European Conference on Computer Vision: Part I*, ECCV '08, pages 304–317, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[10] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. THE MNIST DATABASE of handwritten digits, http://yann.lecun.com/exdb/mnist/. URL `http://yann.lecun.com/exdb/mnist/`.

[11] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions of Information Theory*, 28(2): 129–137, September 2006. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489. URL `http://dx.doi.org/10.1109/TIT.1982.1056489`.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691.

[13] J. Max. Quantizing for minimum distortion. *Information Theory, IRE Transactions on*, 6(1): 7–12, March 1960. ISSN 0096-1000. doi: 10.1109/TIT.1960.1057548.

[14] Louis Joel Mordell. *Diophantine equations*, volume 30. Academic Press, London and New York, 1969.

[15] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '06, volume 2, pages 2161–2168, 2006.

[16] Mohammad Norouzi and David J. Fleet. Cartesian k-means. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3017–3024, 2013.

[17] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[18] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers [speech coding]. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(9):1445–1453, Sep 1988.

[19] C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '08, pages 1–8, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[20] Sivic and Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proceedings of 9^{th} IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477 vol.2, Los Alamitos, CA, USA, October 2003. IEEE.

[21] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *Conference on Neural Information Processing Systems*, pages 1753–1760, 2008.