# Discrete Cosine Transform Locality-Sensitive Hashes for Face Retrieval

Mehran Kafai, *Member, IEEE*, Kave Eshghi, and Bir Bhanu, *Fellow, IEEE*

***Abstract*—Descriptors such as local binary patterns perform well for face recognition. Searching large databases using such descriptors has been problematic due to the cost of the linear search, and the inadequate performance of existing indexing methods. We present Discrete Cosine Transform (DCT) hashing for creating index structures for face descriptors. Hashes play the role of keywords: an index is created, and queried to find the images most similar to the query image. Common hash suppression is used to improve retrieval efficiency and accuracy. Results are shown on a combination of six publicly available face databases (LFW, FERET, FEI, BioID, Multi-PIE, and RaFD). It is shown that DCT hashing has significantly better retrieval accuracy and it is more efficient compared to other popular state-of-the-art hash algorithms.**

***Index Terms*—Discrete Cosine Transform (DCT) hashing, face indexing, image retrieval, Local Binary Patterns (LBP), Locality-Sensitive Hashing (LSH).**

## I. Introduction

FACE retrieval is an important technology used in many different applications, from organizing photo albums to surveillance [1]. In its most common form, a database of images of individuals, one or more images per individual, called the gallery is available. There is also a set of images, called probes, taken of individuals who are to be recognized. The task is to identify which probe and gallery images correspond to the same individual. For example, in the recent event of Boston marathon bombings, images of the suspects taken from street surveillance cameras were matched against government databases to help identify the suspects [2]. In this paper, we assume closed-set retrieval, where every probe individual has one or more matching images in the gallery.

In the most common face retrieval approach, the probe descriptor is compared with each one of the gallery descriptors using an associated distance measure, and the $k$ closest images are retrieved as the answer to the query. While this technique is robust and works well, it requires a linear search of all the gallery images, which can be expensive if the number of gallery images is large.

We investigate the use of locality-sensitive hash functions, applied to the commonly used face descriptors such as Local Binary Patterns (LBP) [3], to generate an indexing structure for nearest neighbor search. This approach has the advantage that no face specific indexable features need to be developed. We show that by using the Discrete Cosine Transform (DCT) hash function [4], excellent retrieval performance can be achieved compared to the baseline, which is to use a linear search. We compare our results with six other hashing methods: LSH [5], $E^2$ LSH [6], KLSH [7], min-hash [8], KSH [9], and K-means codebooks [10], and we show that DCT hashing outperforms the other hash functions on retrieval accuracy and computational efficiency.

A common technique in nearest neighbor search is to use the locality sensitive hashes to find a set of candidates for matching. This set is then matched against the probe using the original distance function. We show that with DCT hashes, a small (order of 50) set of candidates suffices for almost perfect nearest neighbor search. What is more, the size of the candidate set does not need to be increased with gallery size. In fact, retrieval time is dominated by the time taken to generate the hash set of the probe and re-rank the results, which are constant with increasing gallery size. Our experiments show that for practical purposes, retrieval time is near constant with increasing gallery size.

Our goal is to provide an effective algorithm for face retrieval using existing feature descriptors. To this end, we evaluate our approach with three face descriptors: LBP [3], Local Phase Quantization (LPQ) [11], and Histogram of Oriented Gradients (HOG) [12]. The main contribution of this paper is to demonstrate that using the DCT hash function is an effective way to create an index for these descriptors.

In the rest of this paper, Section II discusses the related work and contributions of this work. Section III describes the technical approach. Section IV introduces the face databases and presents the experimental results. Section V concludes this paper and discusses the challenges in DCT retrieval. In the Appendix, we provide the mathematical justification for using the randomized DCT transform for computing the projection vectors. For better understanding, we define the symbols used in this paper in Table I.

## II. Related Work and our Contributions

### A. Related Work

There has been extensive research in image and video retrieval to create index structures that eliminate the need for

Manuscript received July 23, 2013; revised October 14, 2013 and December 29, 2013; accepted January 05, 2014. Date of publication February 11, 2014; date of current version May 13, 2014. This work was supported in part by National Science Foundation grants 0905671 and 0915270. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jing-Ming Guo.

M. Kafai is with Hewlett Packard Laboratories, Palo Alto, CA 94304 USA (e-mail: mehran.kafai@hp.com).

K. Eshghi is with Google Inc., Mountain View, CA 94043 USA (e-mail: kave@google.com).

B. Bhanu is with the Center for Research in Intelligent Systems, University of California at Riverside, Riverside, CA 92521 USA (e-mail: bhanu@cris.ucr.edu)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2014.2305633

TABLE I
DEFINITION OF SYMBOLS

| Symbol | Definition |
|---|---|
| $\alpha$ | common hash suppression factor |
| $\tau, \rho, x, y$ | real number |
| $\beta$ | error rate in expected performance curve |
| $c$ | # of images whose hash set includes hash $h$ |
| $h$ | any single given hash from the hash set of a probe or gallery image |
| $k$ | kernel function |
| $n$ | gallery size |
| $p$ | number of data points forming the kernel matrix |
| $U$ | size of hash universe or number of hash table buckets |
| $A, B$ | zero-centered unit length column vector $\in R^N$ |
| $D_i$ | a unique ID for image $i$ |
| $H$ | number of hashes generated for any probe or gallery image |
| $I$ | reverse index structure |
| $K$ | number of clusters in K-means algorithm |
| $N$ | size of input feature vectors |
| $N_1(x)$ | cumulative distribution function for $\mathcal{N}(0,1)$ |
| $N_2(x, y, \rho)$ | cumulative distribution function of the bivariate standard normal distribution with correlation $\rho$ |
| $Q$ | $U \times N$ matrix of iid $\mathcal{N}(0,1)$ normal random variables |
| $G(A, Q, \tau)$ | hash set of $A$ with respect to $Q$ and $\tau$ |
| $\Pi$ | random permutation of $1, \ldots, U$ |
| $R$ | number of top approximate nearest neighbor results chosen for re-ranking |

linear search [13]. Although, indexing techniques for biometrics [14], [15], [16] and image retrieval [17], [18] have received a lot of attention recently, face retrieval has been less explored.

Chen *et al.* [14] use feature embedding and SVM rank learning for indexing and recognition of 3D objects. Bhanu *et al.* [16] introduce novel features based on minutiae triplets for fingerprint indexing. Dey *et al.* [15] propose to use Gabor energy features for iris data indexing, and to generate a multi-dimensional index key for iris templates. Classical approaches for image retrieval such as k-D trees [19], [20] and R-trees [21] suffer from the curse of dimensionality. Of the more recent approaches, the most notable is the Bag of Words (BoW) technique [22], [23].

Gyaourova *et al.* [24] propose to use reference images for multibiometric indexing. In their approach, an index code is created for the probe and gallery images by comparing them to a set of reference images. An index code is a vector of match scores resulting from the comparison of the gallery/probe image to the reference images. This approach is simple and reduces the retrieval time, however, retrieval is sublinear in the size of the gallery.

Wu *et al.* [25] present a technique where face specific visual words that rely on an identity set are used to create an index structure for face images. Using the index, they retrieve the list of most likely answers to a probe, and then re-rank the results, based on a different descriptor, to find the images most similar to the probe. The re-ranking relies on having a relatively large number of images per individual in the gallery to be reliable. This method is time consuming for large galleries due to the fact that local feature voting has linear complexity.

Retrieval via hashing is commonly used in computer vision applications [26]. Hashing techniques can be divided into two main categorizes; data-independent and data-dependent approaches. A family of data-independent approaches that use random projections for hashing are Locality-Sensitive Hashing (LSH) based methods [6], [27], [28]. The algorithms described in [6], [27] work by mapping each data point to a hash key (a binary-valued vector) by projecting the data points into low-dimensional hamming space. LSH-based methods are commonly used for image retrieval.

LSH methods have a common limitation that they require the data points to have an explicit vector representation. Kulis *et al.* resolve this issue by proposing the Kernelized-LSH [7] which is able to capture the similarity between points without the knowledge of their vector representation by utilizing a single kernel function. In other words, given a kernel function (e.g., Gaussian radial basis function kernel) and a gallery of images, KLSH finds the approximate nearest neighbor to a query image in terms of the kernel function. Min-hash [8] is another LSH-based method used for image retrieval. A min-hash is a single number with the property that two sets have a similar min-hash with the probability equal to their similarity. Min-hashes are grouped into sketches, and near-duplicate candidates are those with a certain number of common sketches. In [29] an algorithm is proposed that computes the min-hash signatures faster than the original min-hash while performing similarly in terms of accuracy.

Data-dependent hashing methods such as semantic hashing [30], geometric hashing [31], kernel-based supervised hashing [9], spherical hashing [32], and codebook-based hash functions (e.g., K-means codebooks [10]) have also been used for image retrieval. They need an expensive preprocessing step such as clustering to create the index structure. For example, K-means codebooks, described in [10], performs K-means clustering of the database descriptors $L$ times, with $K$ on the order of thousands and $L$ on the order of tens. Spectral hashing [33] demands that the hash keys be uncorrelated and balanced, and it overcomes the problem of long binary hash keys; however, for applications with high dimensional data (e.g., face retrieval) it has proven to be ineffective [34]. Heo *et al.* [32] propose using hyperspheres to partition the data for generating the binary codes, rather than using hyperplanes.

Joly *et al.* [17] introduce a data-dependent hashing method that generates independent hashing functions in any kernel space. An important difference between [17] and LSH-based methods is that instead of increasing the collision probability of similar points, the collision probability of dissimilar points is minimized by performing data scattering. He *et al.* [26] propose a hashing algorithm which can perform using any similarity, kernel, and proximity function with compact hash codes. Similar to [17], the proposed approach in [26] focuses on finding independent projections.

Liu *et al.* [9] propose kernel-based supervised hashing, a supervised data-dependent hashing method that effectively learns the kernel-formulated hash functions using the supervised information from the data. Heo *et al.* [32] propose using hyperspheres to partition the data for generating the binary codes, rather than using hyperplanes.

Data-dependent hashing techniques work well for short codes; however they are not as effective when the code length increases [17]. Lin *et al.* [18] propose a hashing method which overcomes this shortcoming by creating similarity preserving sparse representations.

Lu *et al.* [35] investigate the problem of near duplicate image detection (for the purpose of copyright enforcement) through a geometric distortion-resilient image hashing scheme. In their approach, salient points are detected, a mesh is generated on the salient points using Delaunay triangulation, and each triangle is hashed using their proposed distortion resilient hashing algorithm. Unlike their approach, our hash algorithm does not operate on the elements of the image directly; rather, it is applied to the descriptors extracted from the image (e.g., LBP features). We rely on the underlying features (e.g., MLBP [36]) to generate descriptors that are resilient to geometric distortions.

### B. Contributions

The contributions of this paper are:

1) In the appendix, we prove a theorem that justifies the use of the randomized DCT transform for computing the projection vectors. This theorem has applications beyond hashing. It can be used whenever random projections are used in combination with cosine similarity.
2) The analysis framework for collision probability based on the tail probabilities of the bivariate normal distribution is novel. It provides a deeper understanding for the reasons behind the good performance of the hash function.
3) This paper introduces an effective indexing structure for face retrieval using a training-free hash function. It proposes DCT hashing for approximate nearest neighbor face retrieval. The proposed algorithm is compatible with descriptors such as LBP [3], Local Phase Quantization (LPQ) [11], and Histogram of Oriented Gradients (HOG) [12] which have proven to be successful in the field of face recognition.
4) To reduce the computational cost of the proposed algorithm, a histogram-based approach is employed to rank the gallery images rather than using a distance-based approach. Multiple hashes are generated for each image to increase the overlap probability and improve the performance.
5) Our training-free approach is evaluated on a combination of six publicly available face databases to show its ability to perform on face images under constrained and unconstrained settings. We also compare DCT hashing with other hashing algorithms (e.g., LSH, $\mathrm{E}^2$ LSH, KLSH, KSH, K-means codebooks). This comparison shows that DCT hashing not only outperforms other hashing methods in terms of nearest neighbor recall, but also it is so effective that the original distance function is only used for a small percentage of the gallery images (for re-ranking), and yet we achieve results that are essentially identical to linear scan. To the best of our knowledge, there has been no other training-free hash function that can achieve this.

## III. Technical Approach

In the following, we describe the DCT hash function, as a variant of the bivariate normal tail locality-sensitive hash function. We also discuss the distance measure on which it is based, namely the cosine measure. We then present the retrieval framework, and discuss how common hash suppression is used to improve retrieval efficiency and accuracy.

### A. The Bivariate Normal Tail Hash Function

Below we describe the theoretical basis for the Bivariate Normal Tail (BNT) hash function, and discuss how DCT hashing makes BNT computationally tractable. For the purpose of computing the BNT hash, the randomized DCT projection behaves similar to the normal projection in terms of the underlying bivariate normal variables (theorem 1). We provide the mathematical proof and justification in the appendix.

The purpose of a locality-sensitive hash function is to generate, for each input vector, a set of hashes such that if two input vectors are close in the chosen distance measure, their hashes overlap strongly whereas if the two vectors are far from each other, their hashes overlap weakly or not at all. The Bivariate Normal Tail (BNT) hash function uses the cosine between two vectors as the measure of distance between them. The intuition behind the BNT hash function is that the expected size of the overlap between the hash sets of two vectors is directly proportional to the tail probability of a bivariate standard normal distribution with correlation equal to the cosine of the vectors. It turns out that this leads to a very good hash function for information retrieval.

In what follows, an *input vector* is a unit length column vector $\in R^N$.

*Definition 1 (Bivariate Normal Tail Hash Set):* Let $Q$ be a $U \times N$ matrix of iid $\mathcal{N}(0,1)$ normal random variables and $\tau$ a real number. For a given input vector $A$, let $P = QA$. Then

$$G(A, Q, \tau) = \{j : P(j) < \tau\} \tag{1}$$

is the BNT hash set of $A$ with respect to $Q$ and $\tau$.

According to the above definition, the universe from which the hashes are drawn is $1 \ldots U$, where $U$ is the number of rows of $Q$. To be useful, such $U$ must be large, and $\tau$ a negative number far enough from 0 so that the size of the hash set is a small fraction of $U$.

Let $A \in R^N$, $B \in R^N$ be two input vectors where $\cos(A, B) = \rho$. We zero-center the input vectors by subtracting the mean. What we require of our hash function is that the expected overlap between the hash sets of $A$ and $B$ be a monotonically increasing function of $\rho$. Let $P_A = QA$, $P_B = QB$. Let $j$ be an integer, $1 \le j \le U$. Let $v = P_A(j)$ and $u = P_B(j)$. Then it is easy to show that $(u, v)$ is an instance of $\mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$, the bivariate standard normal distribution with correlation $\rho$. It follows that

$$P[u < \tau \wedge v < \tau] = N_2(\tau, \tau, \rho)$$

where $N_2(x, y, \rho)$ denote the cumulative distribution function of the bivariate standard normal distribution with correlation $\rho$. Thus,

$$E(|G(A, Q, \tau) \cap G(B, Q, \tau)|) = N_2(\tau, \tau, \rho)U. \tag{2}$$

In other words, the expected size of the overlap between the hashes of $A$ and $B$ is $N_2(\tau, \tau, \rho)U$.

Let $N_1(x)$ denote the cumulative distribution function for $\mathcal{N}(0,1)$, the standard normal distribution. Now, $N_2(\tau, \tau, 0) = (N_1(\tau))^2$, and $N_2(\tau, \tau, 1) = N_1(\tau)$. Using the techniques described in [37], it is possible to prove that, for any given $\tau$, $N_2(\tau, \tau, \rho)$ is a monotonically increasing function for $\rho$ between

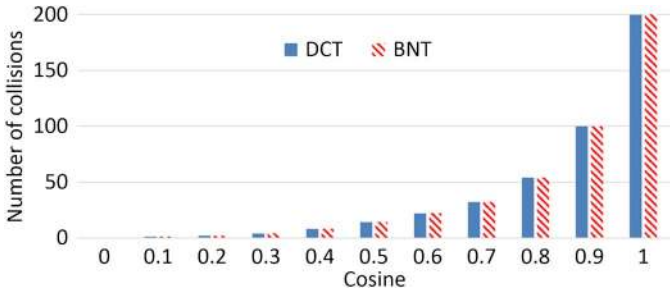


Fig. 1. Collision count comparison between BNT and DCT.

0 and 1. In fact, for $\tau < 2$, its value can be closely approximated by:

$$N_2(\tau, \tau, \rho) \simeq e^{\log(N_1(\tau))(2-\rho)}, \quad (3)$$

i.e., the expected size of the overlap between the hashes of $A$ and $B$ is an exponentially increasing function of their cosine similarity.

As an illustration, for $\tau = -2.7$ and $U = 2^{16}$, the values of the expected overlap between the hash sets (to the nearest integer), as we increase $\rho$ from 0 to 1 in 0.1 increments, is captured by the following sequence: (0, 0) (0.1, 1) (0.2, 3) (0.3, 6) (0.4, 11) (0.5, 19) (0.6, 31) (0.7, 47) (0.8, 72) (0.9, 111) (1, 227).

### B. The Discrete Cosine Transform Function

One of the issues with the BNT function described above is that performing the projection involves $U \times N$ multiplications, which is expensive, since $U$ is large. We use a variant where the projection matrix is a randomized version of the Discrete Cosine Transform matrix, the randomization achieved through permuting the columns of the DCT projection matrix. The advantage of this approach is that computing the projected vector amounts to performing the discrete cosine transform, which takes $O(Ulog_2U)$ operations, a significant improvement on the original function when $N > log_2U$. Also, instead of using a fixed threshold, we sort the projected vector and use the coordinates of the $H$ smallest values as the hash set. This has the advantage that the number of hashes generated per input vector is always the same, which makes the design of the system simpler. We call the hash function based on the DCT projection matrix the DCT hash function.

Figure 1 shows that these modifications of the hash algorithm do not adversely affect the collision properties of the BNT algorithm. The $x$-axis denotes the cosine and the $y$-axis represents the number of collisions (i.e., overlapping hashes). For each value of the cosine on the $x$-axis we generated 2000 pairs of mean-centered unit-length vectors with 1000 dimensions. We computed the number of collisions using the BNT algorithm and the DCT algorithm and plotted the results. Figure 1 demonstrates that the number of collisions for DCT is similar to BNT for all values of the cosine between the vectors. In this experiment $U = 2^{16}$ and $H = 200$.

### C. The DCT Hash Function

The DCT hash function maps input vectors to a set of hashes chosen from a universe $1 \ldots U$, where $U$ is a large integer. We use $H$ to denote the number of hashes that we require per input vector. The input vectors are scaled s.t. their length is $\sqrt{N}$.

TABLE II
COMPUTING THE DCT HASH SET FOR INPUT VECTOR $A$

1) Create a *repeated input vector* $A'$ as follows:
$A' = \underbrace{AAA\ldots A}_{d}\underbrace{000}_{r}$ where $d = U\ div\ N$ and $r = U\ mod\ N$.
*div* represents integer division. We have $|A'| = dN + r = U$.

2) Apply the random permutation $\Pi$ to $A'$ to get permuted input vector $V$.

3) Calculate the discrete cosine transform of $V$ to get $S$.

4) Find the indices of the smallest $H$ members of $S$. These indices are the hash set of the input vector $A$.

TABLE III
MATLAB CODE FOR THE DCT HASH FUNCTION

```
function hashes = DCTHash(input,U,P,H)
% input is the input vector.
% U is the size of the hash universe.
% P is a random permutation of 1:U
%% chosen once and for all and used
%% in all hash calculations.
% H is the desired number of hashes
E=zeros(1,U);
N=length(input);
d=floor(U/N);
for i=0:d-1
    E(i*N+1:(i+1)*N)=input;
end
Q=E(P);
S=dct(Q);
[~,ix]=sort(S);
hashes=ix(1:H);
```

Scaling the input will not change the result because we sort the DCT transforms and pick the top elements.

The hash function takes as input a random permutation $\Pi$ of $1 \ldots U$. Let $A$ be the $N$ dimensional input vector. It should be emphasized that the random permutation $\Pi$ is chosen once and used for hashing all input vectors. Table II shows the procedure for computing the DCT hash set.

Table III presents an implementation of the DCT hash function in Matlab.

### D. The Cosine Measure

The DCT hash function is based on the cosine measure. When cosine is used as a similarity measure, it is advantageous to subtract the population mean from all the vectors. This amounts to moving the center of coordinates to the centroid of the population. We evaluate the effects of mean centering in Section IV-B4. Before subtracting the population mean the center of coordinates is way outside the cluster of the points, resulting in angles between the vectors that are all very small. Thus, as a distance measure, the angle between the vectors is not a very sensitive measure, and the same is true of the cosine of the angle. When the center of origin is moved to the centroid of the data points (by subtracting the population mean) the angles between the vectors become much more differentiated and cosine becomes a more discriminative measure of similarity.

### E. Retrieval Using Hashes

To retrieve the images most similar to the probe image, we construct a reverse index $I$ that maps every hash to the identifiers of the images in which it occurs. Thus, given a hash $h$, the

index returns a list $\{D_1, D_2, \ldots, D_c\}$ of image identifiers. This is the list of the identifiers of all the images whose hash set contains $h$.

To perform nearest neighbor search, the hash function is applied to the descriptor of the probe image, and the resulting hashes are used to look up the entries in the index. This results in $\{L_1, L_2, \ldots L_H\}$, where $H$ is the number of probe hashes, and each $L_i$ is a list of image IDs.

When index $I$ is constructed, the length of the lists for each hash naturally vary; some hashes have more image IDs associated with them than others. An analogy is with keywords and documents: some keywords are more common than others. The more common a hash is, the less informative it is as a means of discriminating between the images for the purpose of similarity search. Moreover, the more common hashes impose an undue computational cost in the retrieval process, since the lists associated with them are long, increasing the cost of computing the histogram. The suppressed hashes are those that occur in many more images than average. To overcome this issue, we perform *Common Hash Suppression (CHS)*.

A threshold is imposed on the length of the ID list that disqualifies some hashes from taking part in the retrieval process. To compute the threshold, the mean $\mu$ and the standard deviation $\sigma$ of the length of the lists associated with all the hashes in the gallery are computed. We then choose $\mu + \alpha\sigma$ as the threshold, where $\alpha$ is a constant. Section IV-B2 shows the impact of $\alpha$ on the retrieval efficiency and accuracy.

Stop words elimination in inverted indexing is a similar concept; however, stop words are determined once and for all texts, whereas common hash suppression is specific to a given collection of images. Common hash suppression is also analogous to TF-IDF [38] in document similarity.

After common hash suppression some of the $L_k$'s are removed. The remaining lists are concatenated. We call this list the retrieval list. Next, a histogram that associates each image ID with its frequency in the retrieval list is created. We call this histogram the retrieval histogram. Thereafter, the histogram is sorted, in descending order, by frequency. This results in a list $D_1, D_2, \ldots$ of image IDs which presents a subset of the gallery images most similar to the query image sorted based on their similarity. In other words, the aforementioned image ID list is the actual final retrieval result; $D_1$ is the most similar image to the query image, $D_2$ is the second most similar image to the query image, and so forth.

Table IV presents the pseudocode for face retrieval using DCT hashes.

TABLE IV
PSEUDOCODE FOR FACE RETRIEVAL USING DCT HASHES

**Parameters**
- $U$: size of hash universe
- $H$: number of hashes per gallery/probe image

**Offline phase**

**Create inverted index of gallery images**
- construct inverted index $I$ with $U$ keys.
- compute $H$ DCT hashes for each gallery image following pseudocode in Table II.
- for each key in $I$ assign gallery image IDs that have that key assigned to them.

**Online phase**

**Generate DCT hashes for probe images and perform retrieval**
- compute $H$ DCT hashes for probe following pseudocode in Table II.
- access buckets of $I$ with keys corresponding to the list of hashes of probe.
- generate list $L : \{L_1, L_2, \ldots L_H\}$ of gallery image IDs from accessed buckets.
- perform common hash suppression.
- count number of occurrences for each gallery image in list $L$.
- generate retrieval histogram, and rank gallery images based on their frequency in the retrieval histogram.

## IV. EXPERIMENTS

We evaluate DCT hashing with various parameter settings and compare it with reference-based indexing [24], linear scan, LSH [5], $E^2$ LSH [6], KLSH [7], KSH [9], min-hash [8], and K-means codebooks [10]. For better comparison, both ranked retrieval rate and Nearest Neighbor (NN) recall are used to evaluate the results. Rank-k retrieval accuracy represents the probability that the correct answer is within the top-k retrieved images. When computing the NN recall, the output of the NN classifier is the ground-truth.

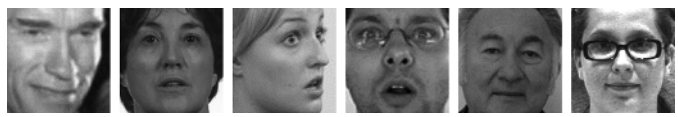

Fig. 2. Sample images from LFW [44], FERET [40], RaFD [42], BioID [43], FEI [41], and Multi-PIE [39].

### A. Data

Experiments were performed with face images taken from six publicly available face databases: Multi-PIE database [39], Facial Recognition Technology (FERET) database [40], FEI database [41], Radboud Faces Database (RaFD [42]), BioID database [43], and Labeled Faces in the Wild (LFW) [44].

The LFW database consists of a total of 13233 images gathered from the web for 5749 people. The FERET face database contains 14126 images of 1199 individuals and is commonly used within the face recognition research community. The FEI face database consists of a total of 2800 images representing 200 individuals (100 males and 100 females) under various poses. The RaFD face database [42] includes 8040 images with various camera angles, expressions, and gaze directions for 67 individuals. The BioID [43] face database contains 1521 images of 23 subjects. The Multi-PIE database [39] contains 755370 images of 337 individuals. Figure 2 illustrates sample face images from all six databases.

Following face detection [45], the face images are cropped and resized to $105 \times 105$ pixels. Each image is divided into 49 regions with a size of $15 \times 15$ pixels. For each region the uniform LBPs with a radius of 1 and 8 neighboring pixels are computed, and a histogram of 59 possible patterns is generated. Histograms of all 49 regions are concatenated in a single histogram of length 2891 to describe the face image.

The setup of the experiments is as follows:

- Probes: 200 images are randomly chosen from LFW, FERET, FEI, BioID, and RaFD databases.
- Gallery: 39500 images from LFW, FERET, FEI, BioID, and RaFD are utilized as the gallery.

TABLE V
IMPACT OF HASH COUNT ON RETRIEVAL PERFORMANCE

| hash count (H) | 300 | 200 | 100 | 50 |
|---|---|---|---|---|
| top-30 retrieval accuracy | 91.4% | 89.5% | 88.5% | 88% |
| histogram length ratio | 60% | 42% | 20% | 9% |

- The random partitioning of the images into the gallery and probe set is conducted 10 times and the average results are reported in the plots.
- The experiments on scalability (Section IV-B6) includes roughly 800,000 images from all six databases.
- An HP mobile workstation with Intel i7 processor and 8 GB of RAM was used to perform all the experiments.

The population mean, estimated by finding the mean of all the gallery descriptors, is subtracted from all the gallery and probe descriptors. The resulting set of vectors is used for computing the DCT hash. For measuring Chi-square and Euclidean distances the original LBP descriptors are used.

### *B. Evaluation Metrics and System Evaluation*

In the following, we investigate how common hash suppression, re-ranking, mean centering, and the number of hashes per descriptor affect the retrieval results. Also, DCT hashing is compared with linear search and other hash algorithms. Finally, we show that DCT hashing is compatible with other descriptors such as LPQ [11] and HOG [12].

An important metric that we use for assessing the computational cost for retrieval is the Histogram Length Ratio (HLR). As explained in Section III-E, the key computational step in the retrieval process is to compute the retrieval histogram. The cost of constructing this histogram is proportional to its length, and as the size of the gallery increases, the average size of this histogram also increases. We calculate the average size of the retrieval histogram and divide it by the size of the gallery; this gives a measure of the computational efficiency of the retrieval process that is independent of the size of the gallery. We call this measure the Histogram Length Ratio.

*1) Hash Count Per Descriptor (H):* Table V presents the impact of the parameter $H$, the number of hashes generated per normalized descriptor. For each hash count, the top-30 retrieval accuracy and *Histogram Length Ratio* (HLR) are measured.

Hash counts above 100 increase the cost of retrieval with little improvement in accuracy. As seen later, good results are achieved with $H = 50$ using CHS and re-ranking.

*2) Common Hash Suppression (CHS):* Figure 3 illustrates how varying the *common hash suppression factor* $\alpha$ has a large impact on both the retrieval accuracy and HLR. Significantly, it shows that increasing $\alpha$ beyond a certain point actually decreases retrieval accuracy. From this chart it is clear that the best combination of retrieval accuracy and HLR performance is achieved when $\alpha = 1.5$.

Table VI presents the impact of CHS on retrieval accuracy. $\alpha = \infty$ corresponds to the case where CHS is not performed. The results show that CHS improves retrieval accuracy for all experimented values of $H$.

Table VII shows the impact of CHS on histogram length ratio. Again, using CHS significantly improves histogram length ratio.

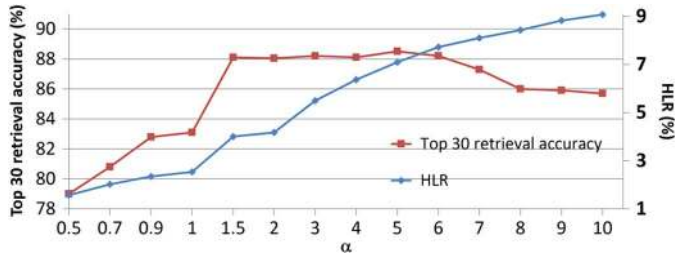


Fig. 3. Histogram Length Ratio (HLR) and top-30 retrieval accuracy for different values of $\alpha$, with $H = 50$ and $n = 11000$.

TABLE VI
IMPACT OF CHS FACTOR $\alpha$ ON TOP-30 RETRIEVAL ACCURACY

| hash count (H) $\rightarrow$ | 300 | 200 | 100 | 50 |
|---|---|---|---|---|
| $\alpha = \infty$ | 91.42% | 89.51% | 88.48% | 88.04% |
| $\alpha = 1.5$ | 91.91% | 90.20% | 89.02% | 88.10% |

TABLE VII
IMPACT OF $\alpha$ ON HISTOGRAM LENGTH RATIO (HLR)

| hash count$\rightarrow$ | 300 | 200 | 100 | 50 |
|---|---|---|---|---|
| $\alpha = \infty$ | 60% | 42% | 20% | 9% |
| $\alpha = 1.5$ | 40% | 27% | 13% | 3% |

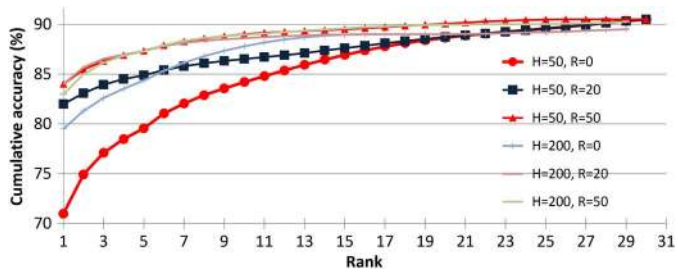


Fig. 4. Impact of re-ranking. $H$ is the hash count, and $R$ is number of re-ranked images ($R = 0$ means no re-ranking was performed).

*3) Re-Ranking:* It is common in approximate nearest neighbor search to re-rank the top $R$ results returned by the approximate solution using the original distance measure [6]. This improves the retrieval accuracy, and when $R$ is small, imposes only a minimal fixed cost on the retrieval process. The Cumulative Match Characteristic (CMC) curve in Figure 4 depicts the impact of re-ranking on retrieval accuracy. $R$ is a constant, $R \ll n$ where $n$ is the gallery size, and Chi-square distance is used for re-ranking. For smaller values of $H$ re-ranking has a dramatic effect.

*4) Mean Centering:* Figure 5 illustrates how mean centering affects retrieval accuracy. For both $H = 50$ and $H = 200$, the top-**30** retrieval accuracy with mean centering is almost the same on average as without using mean centering. Figure 6 shows, for the same experiment, the corresponding HLR values. For both hash values of $H$, HLR decreases dramatically. In other words, the hashes are more distinctive; therefore, fewer hashes are required to obtain the same retrieval accuracy. This is due to how mean centering increases the discriminating power of the cosine similarity measure. Overall, performing mean centering on the feature vectors before generating the hashes, results in smaller retrieval histograms; thus, a more efficient indexing structure.

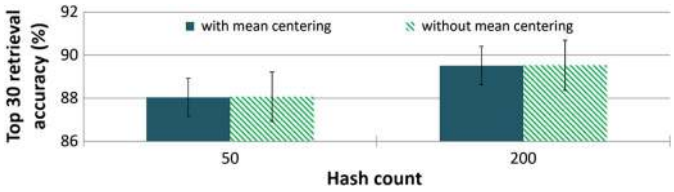


Fig. 5. Impact of mean centering on top-30 retrieval accuracy percentage.

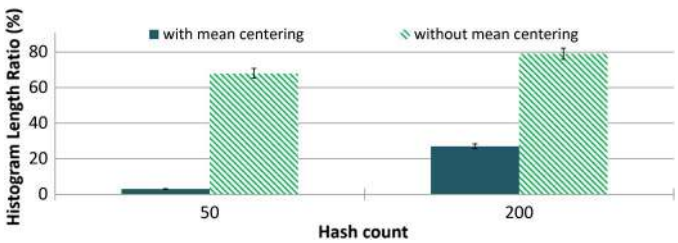


Fig. 6. Impact of mean centering on histogram length ratio.

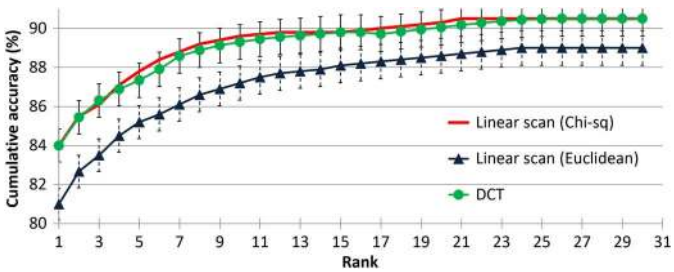


Fig. 7. DCT retrieval vs. linear scan.

TABLE VIII
HISTOGRAM LENGTH RATIO FOR DIFFERENT GALLERY SIZES

| gallery size | 1k | 5k | 10k | 50k |
|---|---|---|---|---|
| HLR | 0.06 | 0.06 | 0.05 | 0.04 |
| gallery size | 100k | 200k | 400k | 800k |
| HLR | 0.04 | 0.04 | 0.04 | 0.03 |

*5) Comparison with Linear Scan:* We compare the accuracy of DCT hash-based retrieval with linear scan, which is our gold standard. For linear scan, the results are shown (Figure 7) with two distance measures: Chi-square and Euclidean. For DCT hashing the parameters are chosen as $H = 50$, $R = 50$, and $\alpha = 1.5$. The CMC curves in Figure 7 present the results.

Figure 7 shows that DCT hashing, augmented with a small, fixed number of Chi-square comparisons (used in re-ranking), can achieve retrieval accuracy that is very close to the best result with linear scan. Linear scan using Chi-square needs to calculate the distance for each one of the gallery images.

These results confirm that using the DCT hash function, combined with common hash suppression and re-ranking, is a significantly more efficient way of performing similarity-based retrieval without having to sacrifice retrieval accuracy.

*6) Scalability:* Table VIII presents the average HLR per probe for different gallery sizes.

The HLR is almost constant with the increase of gallery size, and the cost of processing the histogram is very small, involving one addition for each member of the retrieval list. That is why the cost of DCT retrieval is almost constant with gallery size; the other costs (generating the LBP of the probe image, computing the DCT hash set, re-ranking) dominate the cost of creating the histogram. Even with a gallery of 1,000,000 images, the average size of this list would be roughly 30,000, i.e., only 30,000 additions are required to construct the histogram. This is far below the number of operations required for preprocessing and feature extraction.

TABLE IX
TOP-30 RETRIEVAL ACCURACY FOR DIFFERENT GALLERY SIZES

| method↓ gallery size→ | 1k | 5k | 10k | 50k |
|---|---|---|---|---|
| Linear scan - Chi sq | 92.9% | 92.8% | 92.6% | 90.3% |
| DCT (this paper) | 91.5% | 91.3% | 91.2% | 90.1% |
| method↓ gallery size→ | 100k | 200k | 400k | 800k |
| Linear scan - Chi sq | 89.0% | 87.2% | 85.4% | 84.3% |
| DCT (this paper) | 88.8% | 87.0% | 85.1% | 84.3% |

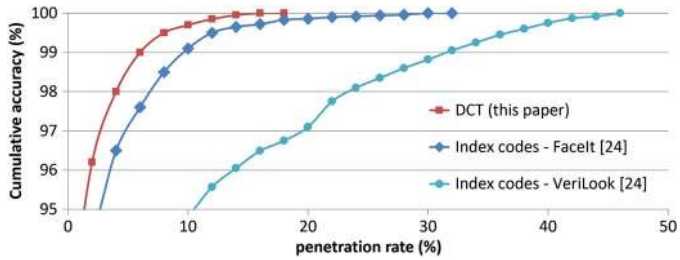


Fig. 8. DCT retrieval vs. reference-based indexing [24].

Table IX shows how retrieval accuracy changes as the gallery size increases. The accuracy for both linear scan and DCT retrieval slightly decreases with increasing gallery size; however, the results show that DCT is an accurate approximation for linear scan. These numbers confirm that retrieval via DCT hashing is extremely scalable.

### C. Comparison with Reference-Based Indexing [24]

The proposed DCT LSH-based retrieval is compared with reference-based indexing codes [24] in Figure 8. The $x$-axis represents penetration rate which denotes the average percentage of gallery images that have to be retrieved. Images for this experiments are randomly selected from the FERET database [40]. We adopt a 10-fold cross validation protocol where in each fold 760 individuals are randomly selected. For the index code method two face matchers are utilized, FaceIt [24] and VeriLook [24]. The results show that DCT retrieval has a better performance with lower penetration rate. It's important to note that reference-based indexing requires that a set of reference images are chosen from the test set for the indexing scheme, whereas DCT retrieval does not have such a limitation.

### D. Comparison with other Hash Algorithms

The proposed DCT hashing retrieval algorithm is compared with other hashing algorithms including LSH [5], $E^2$ LSH [6], Kernelized LSH (KLSH) [7], min-hash [8], Kernel-based Supervised Hashing (KSH) [9], and K-means codebooks [10]. Parameters of all the algorithms are tuned so that they return 50 hashes per image, the same number as used for DCT hashing. All hash functions are treated identically: we used common

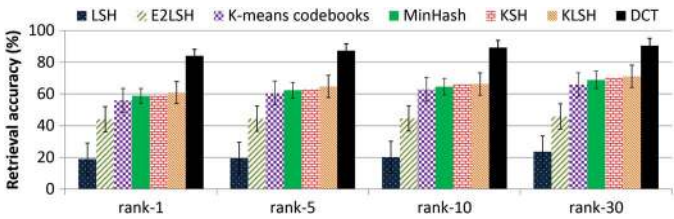


Fig. 9. Rank-k retrieval accuracy comparison for different retrieval methods; LSH [5], $E^2$ LSH [6], KLSH [7], min-hash [8], KSH [9], Codebooks [10], DCT.

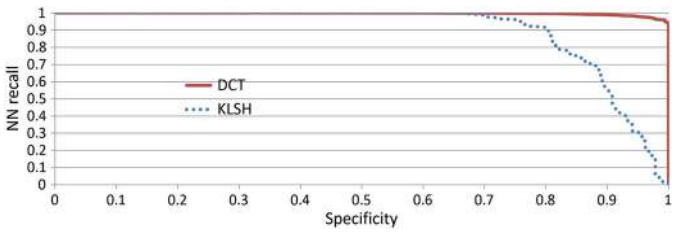


Fig. 10. DCT vs. KLSH approximate nearest neighbor search performance.

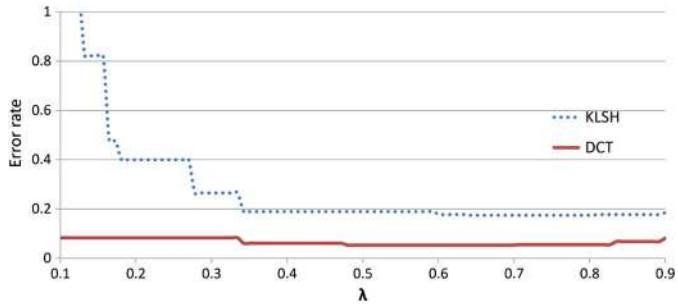


Fig. 11. Expected Performance Curve (EPC); DCT vs. KLSH comparison.

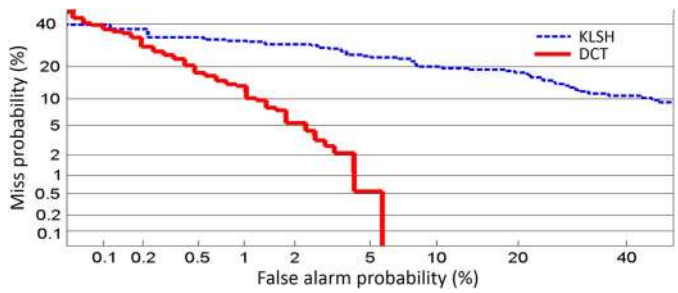


Fig. 12. Detection Error Tradeoff (DET) plot; DCT vs. KLSH comparison.

hash suppression with $\alpha = 1.5$, and re-ranking with $R = 50$. Figure 9 presents the results.

In terms of retrieval accuracy, clearly, DCT hashing is superior to the LSH [5], $E^2$ LSH [6], KLSH [7], min-hash [8], KSH [9], and K-means codebooks [10]. The long binary hash codes generated from multiple hash tables for LSH and $E^2$ LSH have negative impact on the recall rates. On the other hand, if shorter hash codes are generated to increase the recall rates, sub-linearity in term of the gallery size would not be achieved, i.e., a large portion of the gallery would have to processed to achieve acceptable results. In the case of DCT hashing, only one hash table is generated, and multiple hashes for each probe are retrieved; therefore, long binary hash codes are not required to have high recall rates. Also, the cosine measure used in the DCT retrieval algorithm is able to efficiently handle the high dimensional face descriptors without having a negative effect on the nearest neighbor recall rate.

Looking at the Approximate Nearest Neighbor (ANN) problem as a binary classification problem where correct classification (hit) occurs when the nearest neighbor is within the ANN retrieval results, we compare DCT hashing and KLSH (which has the second best results in Figure 9) using multiple evaluation metrics. Figure 10 demonstrates how DCT compares to KLSH in terms of specificity vs. NN recall. Specificity is defined as $1-$ *false alarm ratio*, and NN recall ratio determines how often the nearest neighbor is within the results returned from the hashing algorithm. This simple yet objective plot shows how close a retrieval algorithm is to the baseline nearest neighbor algorithm. Figure 10 shows how DCT hashing outperforms KLSH in ANN search performance.

The Expected Performance Curve (EPC) [46] in Figure 11 compares DCT and KLSH from the viewpoint of tradeoff between false alarm and false reject probabilities. The $x$-axis represents $\lambda \in \mathbb{R}$ where $\lambda \in [0, 1]$. The $y$-axis corresponds to the error rate $\beta$ defined as

$$\beta = \lambda \times \mathrm{FAR} + (1 - \lambda \times \mathrm{FRR}), \tag{4}$$

where FAR is the false alarm ratio and FRR represents the false reject ratio. DCT hashing reports lower error rate compared to KLSH. More importantly, the error rate is almost constant for all values of $\lambda$.

Figure 12 presents the Detection Error Tradeoff (DET) plot comparing the decision error rate of DCT vs. KLSH. The performance is characterized by the miss and false alarm probabilities. Both $x$ and $y$ axes are scaled non-linearly by their standard normal deviates such that a normal Gaussian distribution will plot as a straight line. The results show that DCT hashing reports less miss probability with equal false alarm probability compared to KLSH.

### *E. Other Feature Descriptors*

In all the previous experiments we used LBP as the primary descriptor, and we evaluated DCT hashing based on LBP descriptors. In the following we show that DCT hashing is not a descriptor-specific indexing method and it performs well with other descriptors such as Local Phase Quantization (LPQ) [11] and Histogram of Oriented Gradients (HOG) [12]. LPQ has been adopted for blurred face recognition [11]. For LPQ we use the parameter values $M = 7$, $\alpha = \frac{1}{7}$, and $\rho = 0.9$. For HOG, which has been successfully utilized for face recognition [12], we divide the images into 9 blocks and set the number of orientation bins as 15.

Figure 13 shows how various hashing algorithms perform with different descriptors. The $y$-axis represents the area under NN recall vs. specificity plot (Figure 10). The area has a value between 0 and 1, with values closer to 1 meaning that retrieval is closer to the nearest neighbor classifier in terms of accuracy. In other words, a higher $y$-axis value means a more accurate approximate nearest neighbor algorithm.

Figure 13 shows that DCT hashing has superior performance compared to the other hashing algorithms for LBP, LPQ, and HOG descriptors. Also, DCT performs almost equally well for

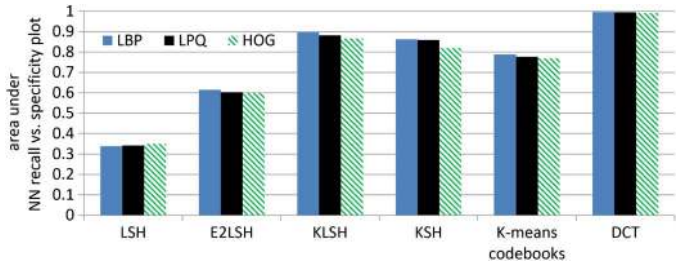


Fig. 13. Approximate nearest neighbor performance using various descriptors.

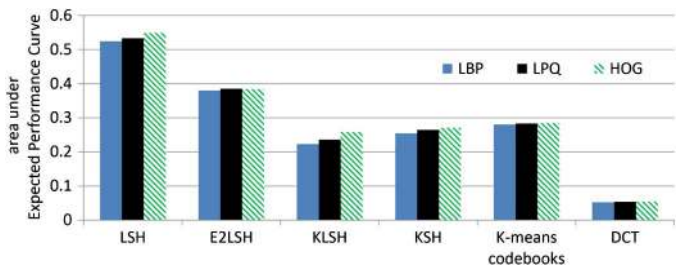


Fig. 14. Expected Performance Curve (EPC) error rate for various descriptors.

LBP, LPQ and HOG. It is important to note that this plot does not say that LBP, LPQ, and HOG have close retrieval results; it shows that DCT hashing is an accurate ANN algorithm for each descriptor regardless of how they perform in terms of accuracy.

Figure 14 evaluates how various hashing algorithms perform with LBP, LPQ, and HOG in terms of area under the expected performance curve. The $y$-axis has values between 0 and 1, with smaller values entailing smaller error rates. Similar to Figure 13, DCT performs equally well for LBP, LPQ, and HOG by reporting almost equal small error rates. Both Figure 13 and Figure 14 show that DCT, similar to LSH, $\mathrm{E}^2$ LSH, KLSH, KSH, and K-means codebooks, is not a descriptor-specific hashing algorithm.

### F. Complexity and Processing Time

*1) Complexity:* Computing the hashes with the proposed DCT hash function is in the order of $U\ log(U)$, where $U$ is the size of the hash universe. $U$ is a constant (in this paper 64 K); thus, the query preparation cost is close to constant. The most costly computational step in KLSH, however, is to compute the matrix $k^{-\frac{1}{2}}$, over the kernel function $k$, which is $\mathcal{O}(p^3)$, with $p$ being the number of data points forming the kernel matrix. In order for KLSH to have sublinear retrieval time $p \ll n$ should hold which decreases the NN recall ($n$ is the size of the gallery). From this point of view, DCT retrieval tends to be computationally cheaper.

Memory usage also affects the retrieval complexity. Assuming that the DCT hash table fits in the main memory, the space complexity is $\mathcal{O}(U \times n)$. Typically, all LSH-based hash functions have a similar space complexity.

*2) Processing Time:* Processing time consists of offline and online processing time. Offline processing time includes the time to firstly generate the hashes for gallery images and secondly insert the generated hashes into the hash table. Generating a hash set with 50 hashes for each gallery image takes 9 ms. In total, generating hashes for the entire gallery with 39500 images takes approximately 6 minutes, and inserting the hashes into the hash table takes 7 minutes, which results in a total of 13 minutes of offline processing for the entire gallery. These results were obtained with non-parallel Matlab and C + + code on an HP mobile workstation with Intel i7 processor and 8 GB of RAM.

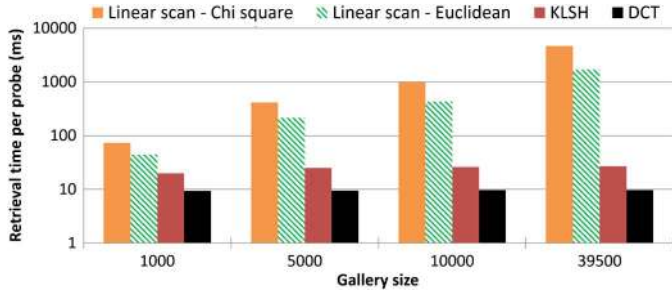


Fig. 15. Retrieval time per probe for different gallery sizes.

We have also addressed the question–how does the retrieval time (online processing time) scale with the size of the gallery? In order to answer this question, we ran experiments with different gallery sizes to measure the average time (in milliseconds) it took to answer a query (not including the time required for preprocessing and computing the LBP). We evaluated the following four methods: linear scan with Chi-square, linear scan with Euclidean, KLSH (Gaussian RBF kernel function, $\sigma = 30$, $b = 300$ bits, 30 Gaussian approximation elements [7]), and DCT hashing ($H = 50, R = 50, \alpha = 1.5$.) Figure 15 shows the results (note that the y-axis is logarithmic). The DCT retrieval time includes the time needed to create the hash set, create the histogram and sort the results, and perform re-ranking. LSH and $\mathrm{E}^2$ LSH are not evaluated due to their poor results in terms of retrieval rate. $K$-means codebooks requires clustering of the gallery descriptors with $K$ in the order of thousands, an extremely expensive procedure, making it unsuitable for practical face retrieval application with large gallery sizes; thus, it is not included in this experiment.

As expected, the time for linear scan increases linearly with the size of the gallery. The time for DCT retrieval, however, is essentially constant, around 9 milliseconds per probe and approximately 11 milliseconds faster than KLSH.

## V. Conclusions

We demonstrated that using DCT retrieval in combination with LBP, LPQ, or HOG descriptors is scalable, with results that are close to Chi-square linear scan and essentially constant query processing time. The proposed DCT hashing algorithm demonstrated superior performance compared to LSH, $\mathrm{E}^2$ LSH, KLSH, min-hash, KSH, and K-means codebooks. Our experiments showed that current LSH-based methods (e.g., KLSH) do not lend themselves well to face retrieval and perform poorly in terms of NN recall rate. We demonstrated how DCT hashing in conjunction with common face descriptors (e.g., LBP, LPQ, and HOG) can be used to create a high performing indexing structure for accurate approximate nearest neighbor search for face images.

DCT hashing is fast and computationally inexpensive; retrieval time is constant (approximately 9 milliseconds per probe) for any size gallery. The experiments were performed on an HP mobile workstation with Intel i7 processor and 8 GB of RAM using images from six publicly available face databases;

FERET, FEI, RaFD, LFW, Multi-PIE, and BioID. The most challenging issue is the cost of computing the hash, which is $O(U\,log(U))$ for the algorithm provided in the paper, where $U$ is the size of the hash universe.

## APPENDIX

We start by defining the symbols used in the statement of the theorem and the proof. All the symbols and definitions used in the appendix have local scope. We then present the main theorem in theorem 1.

*Definitions:*

- A vector is zero centered if the sum of its elements is zero.
- $Q, S \in R^N$. We assume that $Q$ and $S$ are zero centered. Further, we assume that $\|Q\| = \|S\| = \sqrt{N}$
- $\rho = \cos(Q, S)$
- $U$ is an integer greater than $N$. We prove our results in the limit as $U \to \infty$
- $A, B \in R^U$ are repeated vectors of $Q$ and $S$. See table II for the definition of repeated vector.
- DCT is the Discrete Cosine Transform defined as

$$y(\kappa) = w(\kappa) \sum_{i=1}^{U} x(i) cos\left(\frac{\pi(2i-1)(\kappa-1)}{2U}\right) \tag{5}$$

where $\kappa = 1, 2, \ldots, U$ and

$$w(\kappa) = \begin{cases} \sqrt{\frac{1}{U}} & \text{if } \kappa = 1 \\ \sqrt{\frac{2}{U}} & \text{if } 2 \le \kappa \le U \end{cases} \tag{6}$$

- $D \in R^{U \times U}$ is the transformation matrix of the DCT.
- $D_{i.}$ is the $i^{th}$ row of $D$, $D_{.i}$ is the $i^{th}$ column of $D$
- $\widehat{A} = DA$, i.e., $\widehat{A}$ is the DCT of $A$.
- $\widecheck{E} = D^{-1}E$, i.e., $\widecheck{E}$ is the inverse DCT of $E$.
- $k$ is an integer less than $U$. $k$ stays constant as $U \to \infty$
- $K \subset \{2, \ldots, U\}$, $|K| = k$, i.e., $K$ is a set of integers between 2 and $U$ whose size is $k$.
- $\Pi$ is a random permutation of $1, \ldots, U$ chosen uniformly.
- $r = \sum_{i=1}^{U} A(i)B(i)$ is the inner product of $A$ and $B$.

*Theorem 1: As $U \to \infty$ , the set of pairs*

$$\left\{\left(\widehat{A(\Pi)}(i), \widehat{B(\Pi)}(i)\right) : i \in K\right\} \tag{7}$$

*tend to a set of independent, identically distributed bivariate normal random variables with*

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} and\, \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \tag{8}$$

*Proof:* By lemma 7, as $U \to \infty$, The distribution of the variables in

$$\Omega = \left\{\widehat{A(\Pi)}(i), \widehat{B(\Pi)}(i) : i \in K\right\} \tag{9}$$

tends to multivariate normal. Since a multivariate normal distribution is completely determined by its mean and correlation matrix, we will next calculate the mean and correlations.

We note that all rows of $D$ except the first row are zero centered and unit length. Let $v \in \Omega$. Then for some $2 \le i \le U$, $v = D_{i.}A(\Pi)$ or $v = D_{i.}B(\Pi)$ Thus by lemma 11 $E(v) = 0$.

Let $X, Y \in R^U$ be both zero centered. Let $2 \le i, j \le U$. Let $c_i = D_{i.}$ and $c_j = D_{j.}$ Then by lemma 4,

$$E\left[c_i^T X(\Pi) c_j^T Y(\Pi)\right] = \frac{X^T Y}{U-1} c_i^T c_j \tag{10}$$

Since $D$ is an orthogonal matrix, if $i \ne j$ $c_i^T c_j = 0$, i.e., the right hand side of eq. (10) is 0. This means that for all pairs of variables in $\Omega$ where $i \ne j$, their covariance is zero. Since $\Omega$ is multivariate normal in the limit, covariance of 0 means that in the limit these variables are independent.

When $i = j$, $c_i^T c_j = 1$ by orthogonality of $D$. Thus, eq. (10) can be rewritten as

$$E\left[c_i^T X(\Pi) c_i^T Y(\Pi)\right] = \frac{X^T Y}{U-1} \tag{11}$$

We state without proof that when $X, Y$ are repetitions of $Q, S$

$$\lim_{U \to \infty} \frac{X^T Y}{U-1} = cos(Q, S) \tag{12}$$

This can be easily proved by definition of $Q$, $S$, and repeated vectors. Thus, by eq. (11) and eq. (12), $E[\widehat{A(\Pi)}(i)\widehat{A(\Pi)}(i)] = \cos(Q, Q) = 1$ and $E[\widehat{A(\Pi)}(i)\widehat{B(\Pi)}(i)] = \cos(Q, S) = \rho$. Thus, the pair $(\widehat{A(\Pi)}(i), \widehat{B(\Pi)}(i))$ is a bivariate normal random variable with $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. ■

*Lemma 1:* Let $i, j, u, v \in \{1, \ldots, U\}$ such that $i \ne j$ and $u \ne v$. Then

$$P[\Pi(u) = i \wedge \Pi(v) = j] = \frac{1}{U(U-1)} \tag{13}$$

*Proof:* For each permutation $\Pi$, there is a unique pair $(x, y)$ such that $\Pi(u) = x \wedge \Pi(v) = y$. For each such pair, $x \ne y$. There are $U(U-1)$ such pairs, and each is equally probable. Therefore, the probability that $x = i \wedge y = j$ is $\frac{1}{U(U-1)}$. ■

*Lemma 2:* Let $u \in \{1, \ldots, U\}$.
Then $E[A(\Pi(u))B(\Pi(u))] = \frac{r}{U}$
*Proof:*

$$\begin{aligned} E[A(\Pi(u))B(\Pi(u))] &= \sum\nolimits_{i=1}^{U} A(i)B(i)P\left[u = i\right] \\ &= \frac{1}{U}\sum\nolimits_{i=1}^{U} A(i)B(i) = \frac{r}{U} \end{aligned}$$

■

*Lemma 3:* Let $u, v \in \{1, \ldots, U\}$, $u \ne v$. Then $E[A(\Pi(u))B(\Pi(v))] = \frac{-r}{U(U-1)}$

*Proof:* We have $E[A(\Pi(u))B(\Pi(v))] =$

$$\sum_{i=1}^{U} \sum_{j=1, j \ne i}^{U} P[i = \Pi(u), j = \Pi(v)] A(i)B(j) \tag{14}$$

By lemma 1 $P[i = \Pi(u), j = \Pi(v)] = \frac{1}{U(U-1)}$. Thus,

$$\begin{aligned} E[A(\Pi(u))B(\Pi(v))] &= \frac{1}{U(U-1)} \sum_{i=1}^{U} \sum_{j=1, j \ne i}^{U} A(i)B(j) \\ &= \frac{1}{U(U-1)} \sum_{i=1}^{U} A(i) \sum_{j=1, j \ne i}^{U} B(j) \end{aligned} \tag{15}$$

$B$ is zero centered; therefore, $\sum_{j=1,j\neq i}^{U} B(j) = -B(i)$. So

$$E[A(\Pi(u))B(\Pi(v))] = \frac{1}{U(U-1)} \sum_{i=1}^{U-1} -A(i)B(i) = \frac{-r}{U(U-1)} \quad (16)$$

■

*Lemma 4:* Let $c, d \in R^U$ be unit length and zero centered, and $X, Y \in R^U$ be zero centered. Then

$$E[d^T X(\Pi) c^T Y(\Pi)] = \frac{r}{U-1} d^T c \quad (17)$$

*Proof:* Let $F = E[X(\Pi)Y(\Pi)^T]$. From lemmas 1 and 2 it follows that

$$\begin{aligned} F &= \begin{pmatrix} \frac{r}{U} & \frac{-r}{U(U-1)} & \cdots \\ \frac{-r}{U(U-1)} & \frac{r}{U} & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \\ &= \frac{r}{U(U-1)} \begin{pmatrix} U-1 & -1 & \cdots \\ -1 & U-1 & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \\ &= \frac{r}{U(U-1)}(UI - J) \end{aligned} \quad (18)$$

where $I$ is the $U \times U$ identity matrix and $J$ is the $U \times U$ matrix of ones. We have

$$\begin{aligned} E[d^T X(\Pi) c^T Y(\Pi)] &= E[d^T X(\Pi) Y(\Pi)^T c] \\ &= d^T E[X(\Pi) Y(\Pi)^T] c \\ &= d^T F c \\ &= \frac{r}{U(U-1)}(U d^T I c - d^T J c) \end{aligned} \quad (19)$$

Since $d^T$ is zero centered and $J$ is all ones, we have $d^T J = (0, 0, \ldots, 0)$ and $d^T J c = 0$. Thus,

$$\begin{aligned} E[d^T X(\Pi) c^T Y(\Pi)] &= \frac{r}{U(U-1)}(U d^T I c) \\ &= \frac{r}{U-1} d^T c \end{aligned} \quad (20)$$

■

*Lemma 5:* Let $H \in R^U$, where at most k elements of $H$ are non-zero, $H(1) = 0$, and $\max_{1 \leq i < U} H(i) \leq 1$. Let $\breve{H}$ be the inverse DCT transform of $H$. Then $H^T D = \breve{H}^T$, $\breve{H}$ is zero centered, $\max_{1 \leq i \leq U} \breve{H}(i) \leq k\sqrt{\frac{2}{U}}$, and $\|H\| = \|\breve{H}\|$.

*Proof:* By definition of inverse transform $D\breve{H} = H$. Transposing both sides, $\breve{H}^T D^T = H^T$. Since $D$ is an orthogonal matrix, $D^T = D^{-1}$. So $\breve{H}^T D^{-1} = H^T$. Multiplying both sides by $D$, $\breve{H}^T D^{-1} D = H^T D$, i.e., $\breve{H}^T = H^T D$.

$\breve{H}$ is zero centered because $H$ is the DCT transform of $\breve{H}$ and $H(1) = 0$.

Since, as we just proved, $\breve{H}^T = H^T D$, for any $1 \leq i \leq U$, $\breve{H}(i) = H^T D.i$ Since for all $1 \leq i, j \leq U$ $D(i,j) \leq \sqrt{\frac{2}{U}}$, at most $k$ elements of $H$ are non-zero, and all elements of $H$ are less than or equal to one, $\breve{H}(i) \leq k\sqrt{\frac{2}{U}}$.

Finally, $D$ is an orthogonal matrix, thus, $\|H\| = \|\breve{H}\|$. ■

*Lemma 6:* Let $\phi, \psi \in R^U$ have at most $k$ non-zero elements each. Further, let $\phi(1) = \psi(1) = 0$, and

$$\alpha = \phi^T \widehat{A(\Pi)} + \psi^T \widehat{B(\Pi)} \quad (21)$$

Then the distribution of $\alpha$ tends to a normal distribution as $U \to \infty$.

*Proof:* We have $\alpha = 0$ in the following cases:

$$\phi = 0 \wedge \psi = 0 \quad (22)$$

$$\phi = \psi \wedge A = -B \quad (23)$$

$$\phi = -\psi \wedge A = B \quad (24)$$

But zero can be considered a normal random variable with zero mean and standard deviation, so if any of we are eqs. (22) to (24), we are done. The rest of this proof deals with the case that none of eqs. (22) to (24) are true. Let $\Sigma = \sqrt{\sum_{i=1}^{U} \phi^2(i) + \psi^2(i)}$. Since eq. (22) is false, $\Sigma \neq 0$. Thus, we can divide the two sides of eq. (21) by $\Sigma$, getting

$$\frac{\alpha}{\Sigma} = \frac{\phi^T}{\Sigma} \widehat{A(\Pi)} + \frac{\psi^T}{\Sigma} \widehat{B(\Pi)}. \quad (25)$$

Let $\xi = \frac{\alpha}{\Sigma}$, $G = \frac{\phi}{\Sigma}$, and $H = \frac{\psi}{\Sigma}$. Then

$$\xi = G^T \widehat{A(\Pi)} + H^T \widehat{B(\Pi)}. \quad (26)$$

We prove that the distribution of $\xi$ tends to normal distribution as $U \to \infty$. This is sufficient to prove the lemma statement, because $\Sigma$ is a constant and $\alpha = \xi\Sigma$.

From the definition of $G$ and $H$ it can be immediately proved that

$$\sum_{i=1}^{U} \left(G^2(i) + H^2(i)\right) = 1. \quad (27)$$

From eq. (26) and lemma 5, by definition of DCT we can derive

$$\begin{aligned} \xi &= G^T D A(\Pi) + H^T D B(\Pi) \\ &= \breve{G}^T A(\Pi) + \breve{H}^T B(\Pi). \end{aligned} \quad (28)$$

By lemma 5 we have $\|G\| = \|\breve{G}\|$ and $\|H\| = \|\breve{H}\|$; thus, from eq. (27) we have

$$\sum_{i=1}^{U} \left(\breve{G}^2(i) + \breve{H}^2(i)\right) = 1 \quad (29)$$

We are going to use Theorem 3 in [47] to prove the lemma statement. Consider the matrix $C \in R^{U \times U}$, defined as

$$C(i,j) = \breve{G}(i) A(j) + \breve{H}(i) B(j) \quad (30)$$

From eq. (28) it follows that $\xi = \sum_{i=1}^{U} C(i, \Pi(i))$. $\breve{G}$ and $\breve{H}$ are zero centered by lemma 5. Also, $A$ and $B$ are zero centered by assumption. Thus, all the rows and columns of $C$ are zero

centered. Therefore, by Theorem 3 in [47], the distribution of $\xi$ tends to normal as $U \to \infty$ if we can prove

$$\lim_{U\to\infty} \frac{U \max_{1\le i,j\le U} C^2(i,j)}{\sum_{i=1}^{U}\sum_{j=1}^{U} C^2(i,j)} = 0 \tag{31}$$

Let $l = \|A\|^2$, $r = A^T B$ and $s = \breve{G}^T \breve{H}$. Define $\delta$ as

$$\delta = \sum_{i=1}^{U}\sum_{j=1}^{U} C^2(i,j) = \sum_{i=1}^{U}\sum_{j=1}^{U} (\breve{G}(i)A(j) + \breve{H}(i)B(j))^2 \tag{32}$$

We know that $\|A\| = \|B\|$ (due to definition of repeated vector, and the fact that $\|Q\| = \|S\|$). We also know from eq. (29) that $\sum_{i=1}^{U}\left(\breve{G}^2(i) + \breve{H}^2(i)\right) = 1$. Thus, by lemma 9

$$\delta = l + 2rs \tag{33}$$

Since $\|A\| = \|B\|$, $\cos(A,B) = \frac{r}{l}$. Thus, $r = l\cos(A,B)$ and by eq. (33) $\delta = l(1 + 2\cos(A,B)s)$. Let $\omega = 2\cos(A,B)s$. Substituting, we get

$$\delta = l(1+\omega) \tag{34}$$

What we show next is that $\omega > -1$. First, we show that $|\omega| \le 1$.

$$|\omega| = 2|\cos(A,B)\|s| \tag{35}$$

But $|\cos(A,B)| \le 1$, and $|s| \le \frac{1}{2}$ by eq. (29) and lemma 8, so $|\omega| \le 1$. The only conditions under which $\omega = -1$ are

$$\cos(A,B) = 1 \wedge r = -\frac{1}{2} \tag{36}$$

or

$$\cos(A,B) = -1 \wedge r = \frac{1}{2} \tag{37}$$

But by lemma 8 and definition of cosine, eq. (36) implies that

$$A = B \wedge \breve{G} = -\breve{H} \tag{38}$$

Similarly, eq. (37) implies that

$$A = -B \wedge \breve{G} = \breve{H} \tag{39}$$

Since $\breve{G} = \breve{H} \leftrightarrow G = H$, and $\breve{G} = -\breve{H} \leftrightarrow G = -H$, eq. (38) is excluded by eq. (23), and eq. (39) is excluded by eq. (24); thus, $\omega \ne -1$. Thus, since $|\omega| \le 1$, we can conclude that $\omega > -1$, i.e. $1 + \omega > 0$. By eq. (34) $\delta = l(1+\omega)$. Therefore, since by lemma 10 $l \to \infty$ as $U \to \infty$, $\delta \to \infty$ as $U \to \infty$. Now we turn our attention to the numerator of eq. (31). Let

$$\gamma = U \max_{1\le i,j\le U} C^2(i,j) \tag{40}$$

We show that for a constant $c$, $\gamma < c$ as $U \to \infty$. By definition

$$C(i,j) = \breve{G}(i)A(j) + \breve{H}(i)B(j) \tag{41}$$

thus,

$$\gamma = U \max_{1\le i,j\le U} (\breve{G}(i)A(j) + \breve{H}(i)B(j))^2 \tag{42}$$

By lemma 5, for all $1 \le i \le U$

$$\breve{G}(i), \breve{H}(i) \le k\sqrt{\frac{2}{U}} \tag{43}$$

Let $o = \max_{1\le i<N} |Q(i)|$ and $p = \max_{1\le i<N} |S(i)|$. Then because $A$ and $B$ are repetitions of $Q$ and $S$, for all $1 \le i \le U$

$$|A(i)| \le o \text{and} |B(i)| \le p \tag{44}$$

Let us replace $\breve{G}(i), \breve{H}(i), A(i), B(i)$ in eq. (42) with the upper bounds of their absolute values from eqs. (43) and (44). This gives us

$$\gamma \le U\left(ok\sqrt{\frac{2}{U}} + pk\sqrt{\frac{2}{U}}\right)^2 \tag{45}$$

Factorizing and simplifying we get $\gamma \le 2k^2(o+p)^2$. Now, eq. (31) can be rewritten as

$$\lim_{U\to\infty} \frac{\gamma}{\delta} = 0 \tag{46}$$

But we proved that $\delta \to \infty$ as $U \to \infty$, and $\gamma \le 2k^2(o+p)^2$. Since $2k^2(o+p)^2$ is a constant, eq. (46) follows. ■

*Lemma 7:* Let $\Omega = \{\widehat{A(\Pi)}(i), \widehat{B(\Pi)}(i) : i \in K\}$. As $U \to \infty$, the distribution of the random variables in $\Omega$ tends to a multivariate normal distribution.

*Proof:* It is sufficient to prove that any linear combination of the variables in $\Omega$ tends to a normal distribution. But a linear combination of the variables in $\Omega$ can be represented as $\alpha = \phi^T\widehat{A(\Pi)} + \psi^T\widehat{B(\Pi)}$ where $\phi, \psi \in R^U$, and

$$\phi(1) = 0, \psi(1) = 0 (\text{since } 1 \notin K) \tag{47}$$

and

$$\#\text{ non-zero elements in } \phi \text{ and } \psi \le |K| \tag{48}$$

We prove in lemma 6, that given eqs. (47) and (48), the distribution of $\alpha$ tends to a normal distribution as $U \to \infty$, which proves the lemma statement. ■

*Lemma 8:* Let $X, Y \in R^U$ such that

$$\sum_{i=1}^{U} X^2(i) + Y^2(i) = 1 \tag{49}$$

Then $|X^T Y| \le \frac{1}{2}$ and $X^T Y = \begin{cases} \frac{1}{2} & \text{iff} X = Y \\ \frac{-1}{2} & \text{iff} X = -Y \end{cases}$

*Proof:* Let $u = \sum_{i=1}^{U} X^2(i)$, $v = \sum_{i=1}^{U} Y^2(i)$, and $z = |X^T Y|$. By Cauchy − Schwarz inequality [48], $z^2 \le uv$ and

$$z^2 = uv \text{ iff for some scalar } \alpha, X = \alpha Y \tag{50}$$

To determine the maximum of $|X^T Y|$, we consider condition eq. (50). Now, if $X = \alpha Y$, $u = \alpha^2 v$. Let $\beta = \alpha^2$. Thus $u = \beta v$ and from eq. (49). we have $u + v = 1$. From these two, we can derive $v = \frac{1}{1+\beta}$ and $u = \frac{\beta}{1+\beta}$. Thus, $uv = \frac{\beta}{(1+\beta)^2}$, and $\frac{\partial(uv)}{\partial\beta} = \frac{1-\beta}{(1+\beta)^3}$. The maximum of $uv$ under condition eq. (50). is achieved when $\beta = 1$. But $\beta = 1$ implies that $u = v = \frac{1}{2}$,

which implies that $X = Y$ or $X = -Y$. But if $X = Y$, $X^T Y = u = \frac{1}{2}$ and if $X = -Y$, $X^T Y = -u = -\frac{1}{2}$ ∎

*Lemma 9:* Let $\breve{G}, \breve{H}, A, B \in R^U$ s.t. $\|A\| = \|B\|$ and $\sum_{i=1}^{U} (\breve{G}^2(i) + \breve{H}^2(i)) = 1$. Then

$$\sum_{i=1}^{U}\sum_{j=1}^{U} \left(\breve{G}(i)\breve{A}(j) + \breve{H}(i)B(j)\right)^2 = l + 2\,sr \qquad (51)$$

where $l = \|A\|^2$, $r = A^T B$ and $s = \breve{G}^T \breve{H}$.

*Proof:* Proof follows from expanding eq. (51). and then summing the components separately. ∎

*Lemma 10:* Let $Q \in R^N$ such that $\|Q\|^2 = N$. Let $A \in R^U$ be the repetition of $Q$. Let $l = \|A\|^2$. Then as $U \to \infty, l \to \infty$.

*Proof:* Since $A$ is a repeated vector, $\|A\|^2 = \tau\|Q\|^2 = \tau N$, where $\tau$ is the integer division of $U$ and $N$. But $\tau \to \infty$ as $U \to \infty$, proving the lemma statement. ∎

*Lemma 11:* Let $A, c \in R^U$ s.t. $c$ is zero centered. Then $E[c^T A(\Pi)] = 0$.

*Proof:* Let $\overline{A} = \frac{1}{U}\sum_{i=1}^{U} A(i)$. Then $E[c^T A(\Pi)] = \sum_{i=1}^{U} c(i)E[A(\Pi(i))] = \overline{A}\sum_{i=1}^{U} c(i) = 0$. ∎

## REFERENCES


[1] A. Jain, B. Klare, and U. Park, "Face matching and retrieval in forensics applications," *IEEE Multimedia*, vol. 19, no. 1, Jan. 2012, 20 pp.

[2] The New York Times [Online]. Available: www.nytimes.com/2013/04/19/us/fbi-releases-video-of-boston-bombing-suspects.html.

[3] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, "Local binary patterns and its application to facial image analysis: A survey," *IEEE Trans. Syst., Man, Cybern. C*, vol. 41, no. 6, pp. 765–781, Nov. 2011.

[4] K. Eshghi and S. Rajaram, "Locality sensitive hash functions based on concomitant rank order statistics," in *Proc. KDD*, New York, NY, USA, 2008, pp. 221–229, ACM.

[5] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. ICVLD*, 1999.

[6] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, Jan. 2008.

[7] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.

[8] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: Min-hash and tf-idf weighting," in *Proc. BMVC*, 2008.

[9] W. Liu, J. Wang, R. Ji, Y. G. Jiang, and S. F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, 2012.

[10] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recognit. Lett.*, vol. 31, no. 11, pp. 1348–1358, Aug. 2010.

[11] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila, "Recognition of blurred faces using local phase quantization," in *Proc. ICPR*, Dec. 2008.

[12] O. Déniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognit. Lett.*, vol. 32, no. 12, pp. 1598–1603, 2011.

[13] X. Cheng and L. T. Chia, "Stratification-based keyframe cliques for effective and efficient video representation," *IEEE Trans. Multimedia*, vol. 13, no. 6, pp. 1333–1342, Dec. 2011.

[14] H. Chen and B. Bhanu, "Efficient recognition of highly similar 3D objects in range images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 172–179, Jan. 2009.

[15] S. Dey and D. Samanta, "Iris data indexing method using Gabor energy features," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 4, pp. 1192–1203, Aug. 2012.

[16] B. Bhanu and X. Tan, "Fingerprint indexing based on novel features of minutiae triplets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 616–622, May 2003.

[17] A. Joly and O. Buisson, "Random maximum margin hashing," in *Proc. CVPR*, 2011.

[18] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, "Compressed hashing," in *Proc. CVPR*, 2013.

[19] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *Proc. CVPR*, Jun. 2012, pp. 111–118.

[20] T. Vikram, K. Chidananda Gowda, D. Guru, and S. Urs, "Face indexing and retrieval by spatial similarity," in *Proc. Congr. Image and Signal Processing, CISP*, May 2008, vol. 1, pp. 543–547.

[21] P. J. Grother, G. T. Candela, and J. L. Blue, "Fast implementations of nearest neighbor classifiers," *Pattern Recognit.*, pp. 459–465, 1997.

[22] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. ECCV*, 2008.

[23] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. CVPR*, 2007.

[24] A. Gyaourova and A. Ross, "Index codes for multibiometric pattern retrieval," *IEEE Trans. Inf. Forensics Security*, vol. 7, pp. 518–529, Apr. 2012.

[25] Z. Wu, Q. Ke, J. Sun, and H. Y. Shum, "Scalable face image retrieval with identity-based quantization and multireference reranking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 10, pp. 1991–2001, Oct. 2011.

[26] J. He, R. Radhakrishnan, S. F. Chang, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time," in *Proc. CVPR*, 2011.

[27] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. Cambridge, MA, USA: MIT Press, 2006.

[28] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *Proc. NIPS*, 2009.

[29] O. Chum and J. Matas, "Fast computation of min-hash signatures for image collections," in *Proc. CVPR*, 2012.

[30] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2007, vol. 11.

[31] V. D. Kaushik, A. K. Gupta, U. Jayaraman, and P. Gupta, "An efficient indexing scheme for face database using modified geometric hashing," *Neurocomput.*, 2012.

[32] J. P. Heo, Y. Lee, J. He, S. F. Chang, and S. E. Yoon, "Spherical hashing," in *Proc. CVPR*, 2012.

[33] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008.

[34] J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. CVPR*, Jun. 2010.

[35] C. S. Lu and C. Y. Hsu, "Geometric distortion-resilient image hashing scheme and its applications on copy detection and authentication," *Multimedia Syst.*, vol. 11, no. 2, pp. 159–173, 2005.

[36] R. Davarzani, K. Yaghmaie, S. Mozaffari, and M. Tapak, "Copy-move forgery detection using multiresolution local binary patterns," *Forensic Sci. Int.*, vol. 231, pp. 61–72, 2013.

[37] O. A. Vasicek, "A series expansion for the bivariate normal integral," *J. Computat. Finance*, vol. 1, no. 4, pp. 5–10, 1998.

[38] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," in *Proc. Information Processing and Management*, 1988.

[39] R. Grossa, I. Matthewsa, J. Cohnb, T. Kanadea, and S. Bakerc, "Multi-PIE," *Image Vision Comput.*, vol. 28, no. 5, pp. 807–813, 2010.

[40] P. Phillips, H. Moon, P. Rauss, and S. Rizvi, "The FERET evaluation methodology for face-recognition algorithms," in *Proc. ICPR*, 1997.

[41] C. E. Thomaz and G. A. Giraldi, "A new ranking method for principal components analysis and its application to face image analysis," *Image Vision Comput.*, vol. 28, no. 6, pp. 902–913, 2010.

[42] O. Langner, R. Dotsch, G. Bijlstra, D. Wigboldus, S. Hawk, and A. Van Knippenberg, "Presentation and validation of the radboud faces database," *Cognit. Emotion*, vol. 24, no. 8, pp. 1377–388, 2010.

[43] BioID face database [Online]. Available: http://www.bioid.com/downloads/images.html.

[44] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, Labeled faces in the wild: A database for studying face recognition in unconstrained environments, Univ. Massachusetts, Amherst, MA, USA, Oct. 2007, pp. 07–49, Tech. Rep.

[45] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, pp. 137–154, May 2004.
[46] S. B. Bengio, J. Mariéthoz, and M. Keller, "The expected performance curve," in *Proc. ICML*, 2005, pp. 9–16.
[47] W. Hoeffding, "A combinatorial central limit theorem," *Ann. Math. Statist.*, vol. 22, no. 4, pp. 558–566, 1951.
[48] S. S. Dragomir, "A survey on cauchy-bunyakovsky-schwarz type discrete inequalities," *J. Inequal. Pure Appl. Math.*, vol. 4, no. 3, pp. 1–142, 2003.

**Mehran Kafai** (S'11–M'13) received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2005, the M.Sc. degree in computer science from San Francisco State University in 2009, and the PhD degree in computer science from the Center for Research in Intelligent Systems (CRIS), University of California, Riverside in 2013. His research interests are in computer vision, machine learning, and data mining. He is currently a research scientist at Hewlett Packard Laboratories in Palo Alto, California, USA. His recent research has been concerned with big-data analysis and retrieval.

**Kave Eshghi** is a software engineer at Google. He received his Ph.D. in Computer Science from the Imperial College of Science and Technology in 1986. His early interests were Logic Programming and Artificial Intelligence, but more recently he has made contributions in storage technology, particularly in the area of data deduplication. His other interests include information retrieval, with an emphasis on similarity based retrieval of images. Currently, he is working on language modelling for large vocabulary speech recognition.

**Bir Bhanu** (S'72–M'82–SM'87–F'95) received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA; the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA; and the M.B.A. degree from the University of California, Irvine, CA, USA. He was the founding faculty in the College of Engineering, and the founding Professor of electrical engineering with the University of California (UCR), Riverside, CA, and served as its First Chair from 1991 to 1994. Since 1991, he has been the Director of Visualization, and Intelligent Systems Laboratory, UCR. Since 1991, 2006, and 2008, he has been a Cooperative Professor of computer science, and engineering, bioengineering, and mechanical engineering, respectively, with UCR. He was a Senior Honeywell Fellow with Honeywell Inc., Minneapolis, MN, USA. He has been with the Faculty of the Computer Science, University of Utah, Salt Lake City, UT, USA, and with Ford Aerospace, and Communications Corporation, Newport Beach, CA, USA; French Institute for Research in Computer Science, and Control (INRIA);, and IBM San Jose Research Laboratory, San Jose, CA, USA. He is currently a Distinguished Professor of electrical engineering, and serves as the Founding Director of the Interdisciplinary Center for Research in Intelligent Systems, UCR. In addition, he serves as the Director of the National Science Foundation (NSF) Interdisciplinary Graduate Education, Research, and Training Program in video bioinformatics with UCR. He has been the Principal Investigator of various programs for the NSF, the Defense Advanced Research Projects Agency (DARPA), NASA, the Air Force Office of Scientific Research, the Office of Naval Research, the Army Research Office, and other agencies, and industries in the areas of video networks, video understanding, video bioinformatics, learning, and vision, image understanding, pattern recognition, target recognition, biometrics, autonomous navigation, image databases, and machine-vision applications. He is the coauthor of the books *Computational Learning for Adaptive Computer Vision* (to be published), *Human Recognition at a Distance in Video* (Berlin, Germany: Springer-Verlag, 2011), *Human Ear Recognition by Computer* (Berlin, Germany: Springer-Verlag, 2008), *Evolutionary Synthesis of Pattern Recognition Systems* (Berlin, Germany: Springer-Verlag, 2005), *Computational Algorithms for Fingerprint Recognition* (Norwell, MA, USA: Kluwer, 2004), *Genetic Learning for Adaptive Image Segmentation* (Norwell, MA, USA: Kluwer, 1994), and *Qualitative Motion Understanding* (Norwell, MA, USA: Kluwer, 1992). He is the coeditor of *Computer Vision Beyond the Visible Spectrum* (Berlin, Germany: Springer-Verlag, 2004), *Distributed Video Sensor Networks* (Berlin, Germany: Springer-Verlag, 2011), and *Multibiometrics for Human Identification* (Cambridge, U.K.: Cambridge University Press, 2011). He is the holder of 18 (five pending) U.S. and international patents. He has more than 450 reviewed technical publications, including over 120 journal papers and 43 book chapters. Dr. Bhanu is a Fellow of the American Association for the Advancement of Science, the International Association of Pattern Recognition, and the International Society for Optical Engineering. He has served as the General Chair for the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), the IEEE Conference on Advanced Video and Signal-Based Surveillance, the Association for Computing Machinery/IEEE Conference on Distributed Smart Cameras, the DARPA Image Understanding Workshop, the IEEE Workshops on Applications of Computer Vision (founded in 1992 now Winter Applications of Computer Vision Conference), and the CVPR Workshops on Learning in Computer Vision and Pattern Recognition, Computer Vision Beyond the Visible Spectrum, and Multi-Modal Biometrics. He has been on the Editorial Board of various journals and has edited special issues of several IEEE TRANSACTIONS, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–PART B: CYBERNETICS, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, AND IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. He served on the IEEE Fellow Committee from 2010–2012. He was the recipient of the Best Conference Papers and Outstanding Journal Paper Awards, and the Industrial and University Awards for Research Excellence, Outstanding Contributions, Team Efforts and Doctoral/Dissertation Advisor/Mentor Award.