



Revisiting Gist-PCA Hashing for Near Duplicate Image Detection

Hyunwoo Kim¹ · SungRyull Sohn² · Junmo Kim²

Received: 17 December 2015 / Revised: 21 December 2017 / Accepted: 26 March 2018 / Published online: 9 May 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Throw through many layers?

Abstract

This paper presents a scalable method of near duplicate image detection based on Gist-PCA (principal component analysis) hashing. While most of transform coding methods have been interested in nearest neighbor search with applications to similar image search, we solve a range search problems found in near duplicate detection problems. At first, we argue that the PCA hashing of the Gist descriptor is adequate for near duplicate image detection. Then, we decompose the Gist-PCA binary code into a hash key and a residual binary code for scalability into large-scale datasets. In addition, a multi-block approach is incorporated into the method to deal with strong variations, such as image cropping and border framing. Experimental results show that the proposed method is more accurate and faster than the real-valued Gist descriptor and other nearest neighbor search methods.

Keywords Content-based image retrieval · Near-duplicate image detection · Hashing technique · Large-scale multimedia search

1 Introduction

Near duplicate image detection is a key component of Internet or large-scale image processing and analysis. For visual search engines, the duplicated images are considered image spams [22], and can be regarded as useful data to be mined [24]. More interestingly, copyrighted/offensive contents and spam can be filtered, or same category images can be found by search near duplicate images from the manually pre-confirmed dataset [14, 25].

Finding near duplicate images among millions to billions of image datasets is very challenging as in a web-scale image search, however this is different from a general image search, which finds approximate nearest neighbors. First, near duplicate image detection should answer the true or false question by determining a threshold controlling true/false positives. False positives eliminate valuable images, and false negatives result in insufficient removal

of identical images. Second, more resources are needed for the following main multimedia applications, in which the detection process acts as a pre-filter or underlining sub-module. The image description should be compact for storage and access efficiency, and duplicate detection needs to be very fast and accurate with light memory usage. In addition, it is desirable that the technical solutions are easily adapted to general image search problems [2].

The early near duplicate image detectors were based on the hashing-based approaches adapted from document processing. While image search engines retrieve similar or relevant images in a ranked order by measuring distances in high-dimensional feature space, the near duplicate image detectors have been developed to find perceptually/visually identical images by counting hash key collisions between binary codes. Ke et al. [14] introduced a near duplicate detection system by employing locality-sensitive hashing (LSH) to index local descriptors, but it was only applied to a small dataset. Chum et al. [5, 6] adopted the min-Hash algorithm to local descriptors instead of LSH and extended the scalability by improving efficiency and accuracy. Those local descriptors are robust against viewpoint changes [6] and partial occlusions [12], but the computational cost is very expensive.

Others represent images by global features called signatures or fingerprint in the field of content-based image retrieval, and this is found to be scalable to millions to billions

✉ Junmo Kim
junmo@ee.kaist.ac.kr

¹ Advanced Innovation Center for Intelligent Robots and Systems, Beijing Institute of Technology, Beijing, China

² EE Department, KAIST, Daejeon, Korea

images. Zhang et al. [28] represented an image by the average intensities on a regular grid, and its concatenation is defined as a global image feature. The feature vector is projected by PCA and quantized into a hash code. Their experimental results are shown on 2.5 million images, and similar algorithms was reported to work in billions of images in an image-based annotation system [24]. Douze et al. [7] demonstrated the Gist descriptor, which was originally proposed as a computational model of the recognition of real world scenes [19], is suitable for large-scale near duplicate detection. Their method outperformed the state-of-the-art method, i.e., bag-of-features based on local features, in terms of accuracy, memory efficiency, and search speed. In the bag-of-features image search framework, they quantized the Gist features by k-means clustering, and the residual vectors were embedded into hamming space. Their efficient indexing structure of the Gist descriptor resulted in comparable accuracy to exhaustive search.

In the field of near-duplicate image detection, other visual descriptor are proposed instead of Gist descriptor. [17] proposed a variable-length signature and utilized the earth mover's distance to compare the variable-length signatures. Furthermore, in [29], an efficient coarse-to-fine Riemannian image search strategy was developed to improve efficiency while keeping accuracy.

Recently, learning-based transform coding has been getting attention for approximate nearest neighbor search because it represents images or high dimensional feature vectors as compact binary codes while preserving their neighborhood structure and/or relative distances. Torralba et al. [20, 21] employed supervised machine learning methods based on boosting and the Restricted Boltzman Machines (RBMs) to compress the Gist descriptor into a few hundred bits. Other approaches that have been utilized include unsupervised or semi-supervised machine learning techniques. Those methods mostly depend on PCA for dimensionality reduction and then encode the projected vectors into binary codes using various quantization techniques [3, 8, 13, 23, 26]. In [26], spectral hashing (SH) method assigned more bits to more relevant directions, and in [23], the semi-supervised hashing (SSH) algorithm relaxed the orthogonality constraints of PCA. Jegou et al. [13] showed that applying a random orthogonal transformation to the PCA-projected data works better than SH and SSH. Gong and Lazebnik [8] improved the previous methods by applying an orthogonal transformation, which directly minimizes the quantization error, called iterative quantization (PCA-ITQ). Furthermore, the relationship between nearest neighbor search and signal quantization was investigated by both Chandrasekhar et al. [4] and Brandt [3].

Most of the previous PCA hashing applied to image descriptor focus on similarity preserving property and coding error minimization but do not pay much attention to optimizing the performance in terms of receiver operating characteristic (ROC) curve or precision-recall curve. Near-duplicate image detection is one of the important applications belonging to this category. For example, in an e-commerce service, near-duplicate products should be hidden or grouped in the search results. Also, for content-based image search, near-duplicate images should be grouped with high precision in order to extract shared keywords from surrounding texts. Another example is copyright protection, except original copyrighted images, other near-duplicate images should be prohibited to be exposed in public or search results. In this paper, we present a fast and efficient large-scale system for near duplicate image detection based on PCA hashing. While other transform coding methods are interested in nearest neighbor search for similar image search, we introduce an effective and efficient range search method aiming to solve near duplicate detection problem by revisiting Gist-PCA hashing. Our contributions are threefold:

- We argue that a PCA binary coding of the scene gist is better for near duplicate image detection because a PCA binary embedding can keep as much distance information as a PCA transformation does (Section 2.)
- We propose a scalable method of near duplicate image detection by decomposing a Gist-PCA binary code into a hash key and a residual binary code, for large-scale duplicate image detection (Section 3.)
- The multi-block approach, i.e., encoding cropped image regions as well as a holistic image region, can be further incorporated to effectively deal with strong variations, such as image cropping and border framing (Section 4.)

The paper is organized as follows. Section 2 explains how to encode the global image characteristics based on Gist-PCA hashing. In Sections 3 and 4, we introduce the bit decomposition and multi-block approach for large-scale retrieval and the accuracy improvement, respectively. Experimental results are presented in Section 5, and we conclude in Section 6.

2 Binary Image Representation for Near Duplicate Detection

The near duplicate images are herein defined as the images related by image-to-image transformations, including various photometric and geometric deformations, such as contrast change, jpeg compression, resizing, cropping, border framing, and watermarking. In other words, they are

type	original	dark_50	brightness_50	centercrop_10	jpegcomp_10	border_b_10	watermark_s1a4	centercrop_20	border_b_20
Dist	0	5	6	11	18	24	27	29	31

Figure 1 Simulated image variation for the near duplicates and the Hamming distance from the original image. The first rows are the original image and its near duplicate variations. In the second row, the near duplicate variation name and their hamming distance for the 128-bit binary code are shown.

regarded as perceptually identical images instead of images taken from identical objects/scenes [5, 7]. The examples of near duplicate image variations¹ are presented in Fig. 1, and more details will be given later in Section 5.

To encode the perceptual similarity of images, the Gist descriptor is used. It is originally proposed to capture a set of perceptual dimensions of visual scenes in [19], and recently, it has shown promising results for image search [7]. The Gist descriptor is extracted by concatenating 20 Gabor responses to 4x4 non-overlapping image blocks, and it is represented by a 960-dimensional real-valued vector, for color images. The visual similarity between different images can be measured by computing the Euclidean (L_2) distance in the Gist vector space.

Given a query image I_q , our goal is to search the near duplicate images in the database $\{I_1, \dots, I_N\} \in \mathcal{I}_{ref}$. Their corresponding Gist descriptors are represented by $\mathbf{x}_q \in \mathbb{R}^n$ and $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^n$, respectively, where n is vector dimension. The image similarity between the query \mathbf{x}_q and a database image $\mathbf{x}_i \in \mathcal{X}$ is measured by the L_2 distance, i.e., $d(\mathbf{x}_q, \mathbf{x}_i) = \|\mathbf{x}_q - \mathbf{x}_i\|_2$. The set of near duplicate images \mathcal{X}_q are determined by checking that the Gist descriptor of a database image is located within a certain distance, d_{Gist} , from a query vector, i.e., $\mathcal{X}_q = \{\mathbf{x}_i | d(\mathbf{x}_q, \mathbf{x}_i) \leq d_{Gist}, \mathbf{x}_i \in \mathcal{X}\}$, and the other database images, i.e., the complements in \mathcal{I}_{ref} , outside the range are non-duplicate images: \mathcal{X}_q^c .

¹ They include intensity change (brightness_50, dark_50), blurring (blur_11x11), resizing (resize_h3w3), jpeg compression (jpegcomp_10, jpegcomp_15, jpegcomp_20), image cropping (centercrop_10, centercrop_20, leftcrop_10), border framing (border_w10, border_w20, border_b10, border_b20), and watermarking (watermark_s1a4, watermark_s2a5). Notationally, the first string and the following number denote the variation type and degree, respectively. In border type, b and w mean the black and white border frames, respectively. In cropping, the “centercrop” and “leftcrop” differ in the alignment before the cropping, and the next degree is the cropping ratio in terms of image width and height, which is different from the surface cropping ratio provided in Copydays.

The determination of the distance threshold, d_{Gist} , is critical for the performance of near duplicate detection. When the threshold is too large, non-duplicate images are misclassified as near duplicate images, and when it is too small, many near duplicate images are missed. It can be fixed for all the queries or vary adaptive to each query. For the near duplicate image detection, we found that the advantage of the adaptive threshold is not big after binary quantization, and it is not easy to find the adaptive threshold for a query. The fixed threshold is used in this paper, and it can be determined in the training stage when the allowed false acceptance rate is given.

2.1 Binary Coding

Image Compression
To deal with millions and billions of large-scale images, the size of the real-valued high-dimensional descriptors is too large to store, and the distance computation is too heavy to be instantly performed, so the compression of the descriptor is crucial. For compact binary coding, while other (approximate) nearest neighbor methods only maintains the local neighborhood structure, we try to find a binary embedding that keeps the distance information as much as possible, so that a range query can discriminate near-duplicate and non-duplicate images. In other words, when the near duplicate images are within a certain range in the original vector space, it is expected that they can also be found in a certain range in the quantized space.

An m -bit binary coder, $h : \mathbb{R}^n \rightarrow \{0, 1\}^m$, of a n -dimensional vector is designed to retain the original distance structure as much as possible. In the transform coding framework, the binary coding can be broken into two steps. First, the Gist descriptor ($\mathbf{x} \in \mathcal{X}$) is transformed into a low-dimensional vector space $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$, then the vector are encoded into a binary code by $b : \mathbb{R}^k \rightarrow \{0, 1\}^m$. The first step is called transform coding ($f(\cdot)$) and the next is binary quantization ($b(\cdot)$).

Let $h(\mathbf{x}_q)$ and $h(\mathbf{x}_i)$, be the m -bit binary codes corresponding to the Gist vector of a query image \mathbf{x}_q and

that of a database image $\mathbf{x}_i \in \mathcal{X}$, respectively. We then model the Gist vectors, \mathbf{x}_q and \mathbf{x}_i , as random vectors drawn from an underlying unknown distribution. The sets of the binary codes belonging to the ground truth of near duplicate images and the others are denoted by \mathcal{B}_q and \mathcal{B}_q^c for the query $h(\mathbf{x}_q)$, where $h(\mathbf{x}_i) \in \mathcal{Y} \equiv \{0, 1\}^m$, and $\mathcal{B} = \mathcal{B}_q \cup \mathcal{B}_q^c$. Therefore, a desirable binary coding needs to maximize the range preserving probability:

$$\sum_{i=1}^N P(h(\mathbf{x}_i) \in \mathcal{B}_q | \mathbf{x}_i \in \mathcal{X}_q, \mathbf{x}_q) - \lambda P(h(\mathbf{x}_i) \in \mathcal{B}_q^c | \mathbf{x}_i \in \mathcal{X}_q^c, \mathbf{x}_q), \quad (1)$$

for any query \mathbf{x}_q , where λ is a weighting factor. The binary code maximizes the probability that true duplicates are located within a certain range and at the same time penalizes the probability that non-duplicates are inside the range. In terms of ROC criteria, the first and second terms are corresponding to high detection rate and low false acceptance rate, respectively. While most of the previous hashing methods have been applied to image descriptor focusing on similarity preserving property and coding error minimization, we are optimizing the performance in terms of receiver operating characteristic (ROC) curve in Eq. 1.

2.2 Gist-PCA Hashing

To find the optimal binary coding maximizing of the range preserving probability (Eq. 1), PCA hashing is investigated. Solving Eq. 1 is intractable because of the combinatorial property of binary codes. In the paper, we describe why PCA hashing is a better choice compared to other state-of-the-art hashing methods with the similar preserving property. We will explain it with comparison study and the theoretical analysis for comparison of the performance of PCAH and that of other state-of-art methods.

PCA is a linear transformation that allows coordinate axes to be determined in such a way as to retain as much distance information as possible in a mean-square sense [27]. The principal axes of PCA learns which axes have the largest variations, where the variations effect the L_2 distance (i.e., mean square distance) most [3]. PCA embeds a n -dimensional Gist descriptor \mathbf{x}_i into a k -dim vector space ($k \leq n$.) Given a training set, the Gist vectors are mean centered by subtracting the data mean, denoted by \mathbf{t} , and they concatenate into a $n \times N$ matrix. From the covariance matrix, the eigenvectors with the k -largest eigenvalues construct transform coordinates, which are denoted by $\mathbf{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$.

Using the learned transform matrix \mathbf{W} , PCA hashing is straightforward. First, a mean centered Gist vector $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}} \in \mathbb{R}^n$ is projected into a low-dimensional vector space by PCA: $f(\tilde{\mathbf{x}}_i) = (f_1(\tilde{\mathbf{x}}_i), \dots, f_k(\tilde{\mathbf{x}}_i)) = (\mathbf{w}_1^T \tilde{\mathbf{x}}_i, \dots, \mathbf{w}_k^T \tilde{\mathbf{x}}_i)$, then it is quantized into a binary bit in each dimension:

$b(y_j) = Q(y_j)$, where $y_j = f_j(\tilde{\mathbf{x}}_i)$ and $Q(y_j) = \text{sgn}(y_j)$. Therefore, the binary coding is represented by

$$\mathbf{z}_{PCAH} = h(\mathbf{x}_i) = b \circ f(\tilde{\mathbf{x}}_i) = (Q(\mathbf{w}_1^T (\mathbf{x}_i - \bar{\mathbf{x}})), \dots, Q(\mathbf{w}_k^T (\mathbf{x}_i - \bar{\mathbf{x}}))), \quad (2)$$

for any gist vector \mathbf{x}_i .

This approach is deceptively simple, but it turns out that this approach is better than other state-of-the-art methods in terms of the range preserving probability in Eq. 1. The binary coder in Eq. 2 is only PCA followed by a binary quantizer with a hard threshold. In the perspective of hashing, the Gist vector space is partitioned by the hyper planes orthogonal to the PCA principal axes \mathbf{w}_j , for $j = 1, \dots, k$, which have the most variations.

We assume that the maximization of the range preserving probability requires the preservation of the L_2 distance metric of the original feature space after binary hashing as much as possible. The problem is how to represent the feature vectors using a small number of binary bits. First, the PCA finds the most informative axes. The PCA axis corresponding to the largest eigenvalue will preserve the L_2 distance mostly faithfully because the axis contains most variations. Second, the PCA minimizes the reconstruction error between the original feature and the projection because it retains the maximum variation.

Current best methods

State-of-the-art methods Before comparing PCAH with other state-of-art methods including SH [26], LSH [11], and ITQ [8], the major differences between each methods are investigated. For given n -dimensional gist feature(\mathbf{x}), the feature is translated so that they are of zero mean($\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$). Then, LSH projects \mathbf{y} onto d -random axes, producing d -dimensional feature $\mathbf{y}_{LSH} = \mathbf{y} * (\mathbf{A}_{LSH})$ and quantize each dimension to get final hash code ($\mathbf{z}_{LSH} = \text{sgn}(\mathbf{y}_{LSH})$). PCAH, ITQ, and SH transform \mathbf{y} with PCA to get \mathbf{y}' . PCAH directly quantizes each component of \mathbf{y}' by thresholding at zero($\mathbf{z}_{PCAH} = \text{sgn}(\mathbf{y}')$). ITQ rotates \mathbf{y}' by multiplying an orthogonal matrix to minimize the quantization error, and quantize by thresholding at zero($\mathbf{z}_{ITQ} = \text{sgn}(\mathbf{y}' * \mathbf{A}_{ITQ})$). SH produces d eigen functions which apply equally-spaced thresholds to each component of \mathbf{y}' , resulting in total $d * n$ components, and take d components of smallest eigenvalues. Therefore, each component of \mathbf{y}' can be assigned varying number of bits and each bit can be quantized with many thresholds(note that PCAH, ITQ, and LSH have only one threshold at zero).

2.3 Comparison Study

We experiment on Copydays [13], which is the most popular benchmark for near-duplicate detection. Peekaboo dataset is used for training hash functions. Figure 2a shows a ROC

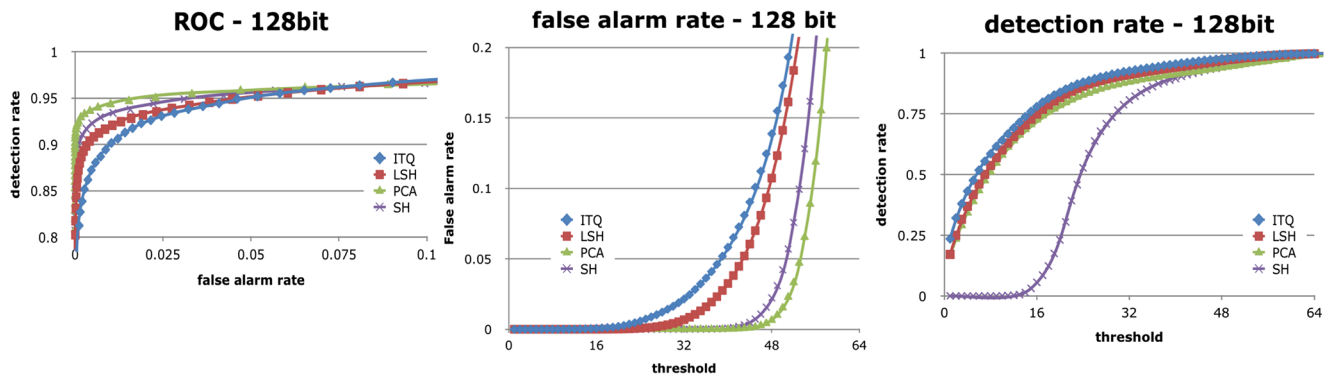


Figure 2 The comparison of ROC, FAR, and DR for the compared methods.

	original	1	2	3	4	5	6	7	8	9
PCA										
Dist		26	29	34	38	46	46	47	47	48
ITQ [8]										
Dist		6	10	18	19	22	26	21	24	24
SH [26]										
Dist		33	37	41	42	44	44	47	50	54
LSH [11]										
Dist		30.87	31.00	31.20	31.87	32.20	34.40	34.73	34.80	34.80

	original	1	2	3	4	5	6	7	8	9
PCA										
Dist		5	22	35	36	41	47	49	53	54
ITQ [8]										
Dist		5	19	23	33	37	39	41	48	52
SH [26]										
Dist		28	37	38	43	47	48	48	48	51
LSH [11]										
Dist		10.20	19.67	31.47	34.73	43.07	49	49.2	53.87	55.87

Figure 3 The example of the distance statistics of near-duplicate images from an original image.

curve of the state-of-the arts methods and the proposed method. As shown in Fig. 2b PCA hashing has a wider range of acceptable thresholds than ITQ and LSH. On the other hand, Fig. 2c shows that PCA hashing has a detection rate larger than the SH for a fixed threshold. This observation will be also theoretically backed up to explore the performance of PCA hashing in a ROC curve.

Also, Fig. 3 shows visual results of the compared methods. the correctly retrieved images, the correctly rejected images, and falsely detected images are marked in cyan, yellow, and magenta, respectively. PCA hashing has a better discriminative power to separate the true positive (correctly retrieved) and true negative (correctly rejected) images.

The first step will show that, for fixed low false alarm rates, the range of the hamming ball size (threshold) for PCA that meets FAR constraints is larger than other methods. In this step, you will see that PCA has smaller variances in hamming distances between arbitrary queries, which results in the larger range of the permitted threshold.

The Relationship between FAR and Its Threshold For arbitrary images A and B, the false alarm rate for threshold γ can be written as

$$\begin{aligned} FAR &= P(d_H(A, B) \leq \gamma | L(A) \neq L(B)) \\ &= P(d_H(q_1, q_2) \leq \gamma), \end{aligned} \quad (3)$$

where q_1 and q_2 are the binary hash codes of arbitrary but distinct query images.

Now, we will focus on the maximum threshold (γ_{\max}) allowable for a fixed low false alarm rate of each hashing method. A typical FAR is in the range of 0.01 – 0.1, which is much smaller than 0.5. Then, the analysis of $d_H(q_1, q_2)$, which is the hamming distance of arbitrary queries \mathbf{q}_1 and \mathbf{q}_2 , is necessary to compare the γ_{\max} of each hashing methods. The key property of $d_H(\mathbf{q}_1, \mathbf{q}_2)$ is that its mean is fixed to $d/2$ for any hashing method that assigns 0 and 1 with equal probability (i.e. mean-centered bits) ($E[d_H(\mathbf{q}_1, \mathbf{q}_2)] = E[\sum_{n=1}^d d_H(\mathbf{q}_{1n}, \mathbf{q}_{2n})] = \sum_{n=1}^d E[d_H(\mathbf{q}_{1n}, \mathbf{q}_{2n})] = \sum_{k=1}^d 0.5 = \frac{d}{2}$). Most hashing methods, including PCAH, SH, LSH, and ITQ, adopt a mean-centering process (translation of data points so that they are of zero mean), which results in the equal probability of 0 and 1 for every bit. Therefore, for the methods we will deal with, we can assume that the mean value of $d_H(\mathbf{q}_1, \mathbf{q}_2)$ is fixed to $d/2$. Next, we consider $\text{Var}(d_H)$ (the variance of $d_H(\mathbf{q}_1, \mathbf{q}_2)$) of each method. If the variance is large, as in the case of ITQ in Table 1, the γ_{\max} s.t. $P(d_H(\mathbf{q}_1, \mathbf{q}_2) \leq \gamma) = FAR$ will be small. This observation simplifies our original purpose of showing that PCAH has minimum variance, and

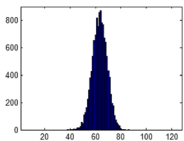
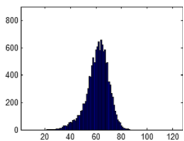
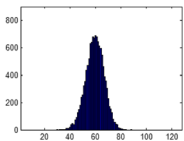
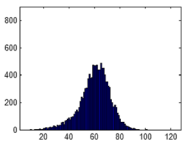
that other methods have variances larger than that of PCAH. This statement strongly supports the notion that PCAH has the largest largest γ_{\max} with an accompanying low false alarm rate.

3 Binary Code Decomposition for Scalability

A PCA projection of the Gist image description is decomposed into a hash key and a residual binary code for fast neighbor search and compact storage, respectively. A PCA-hashing-based binary hashing method of the Gist image descriptor is introduced by decomposing a PCA quantized vector into a hash key and a residual binary code for efficient storage and search. The k -most significant bits of the binary code are used as a hash key for coarse vector quantization, then the retrieved codes are finely measured in Hamming distance for the residual bits. The PCA transform coding followed by quantization can more effectively separate near duplicate images from other visually similar non-duplicate images, compared to other sophisticated coding techniques [3, 8] and even to the uncompressed real-valued Gist descriptor. Also, we show that it is closely related to the bag-of-feature image search architecture with hamming embedding [7].

To retain enough information, the length of the PCA-based binary code will be at least 64 bit, which contains 78.9% of the energy of the data, with 128-bit length where 87.4% is desirable for practical applications. The exhaustive search is too expensive in large-scale datasets, and the hash table size will be too large to fit into memory. In [21], the 30-bit was a practical maximum at a moderate-level of PC memory ($2^{30} \times 8$ bytes/table entry = 8G Bytes) in a single machine. To deal with the longer binary code, we decompose the binary code into a hash key and a residual binary code. The k -bit binary code in Eq. 2 is decomposed into the following two codes: $h(\mathbf{x}_i) = (h_1(\mathbf{x}_i), h_2(\mathbf{x}_i))$, where $h_1(\mathbf{x}_i) = (Q(\mathbf{w}_1^T \mathbf{x}_i - \bar{\mathbf{x}}), \dots, Q(\mathbf{w}_l^T \mathbf{x}_i - \bar{\mathbf{x}}))$, $h_2(\mathbf{x}_i) = (Q(\mathbf{w}_{l+1}^T \mathbf{x}_i - \bar{\mathbf{x}}), \dots, Q(\mathbf{w}_k^T \mathbf{x}_i - \bar{\mathbf{x}}))$, and $l \leq k$, for any Gist vector \mathbf{x}_i . The first code $h_1(\mathbf{x}_i)$ is the hash key, which corresponds to the l -most significant bits of the binary code, and the second code $h_2(\mathbf{x}_i)$ is the residual binary code, which stored for the finer range measure. It can be interpreted that the Gist vector space is divided by the hyperplanes orthogonal to the principal axes of PCA (equivalent to K-means clusters), and then each cluster is further quantized by the residual binary bits. The approach is closely related to the bag-of-feature image descriptor with hamming embedding [7]. The hash key and residual bits correspond to the bag-of-feature quantizer and hamming

Table 1 The comparison of variance of the-state-of-the-art methods.

	PCAH	LSH [11]	SH [26]	ITQ [8]
Mean_dist	63.34	61.36	59.51	60.42
Near_dist	47.38	38.01	40.28	35.13
Variance	34.15	75.14	51.30	135.65
Dist.				

embedding, respectively, but the computational cost is expensive because the bag-of-feature quantizer needs an exhaustive distance comparison of the Gist descriptors.

To build the indexing structure, the l -bit hash key constructs a hash table with a size of 2^l , and the residual codes are stored into the corresponding hash buckets in linked lists. Given a query \mathbf{x}_q , its binary code $h(\mathbf{x}_q)$ is decomposed into $h_1(\mathbf{x}_q)$ and $h_2(\mathbf{x}_q)$. Using the hash key $h_1(\mathbf{x}_q)$, the buckets, including relevant images, are selected when they are located inside the range d_{key} . The buckets are found by selecting all the hash keys within the hamming ball of size d_{key} , and the selected buckets include the candidate items for near duplicate images: $\mathcal{N}_{h_1}(\mathbf{x}_q) = \{\mathbf{x}_i | d_H(h_1(\mathbf{x}_q), h_1(\mathbf{x}_i)) < d_{key}\}$. The buckets can be directly accessed by the address offset by the sum of xor between $h_1(\mathbf{x}_q)$ and the precomputed hamming ball. The distance of the items for each bucket is measured in hamming distance between the full-length binary code: $d_H(h(\mathbf{x}_q), h(\mathbf{x}_i))$. The distance can also be computed by $d_H(h_1(\mathbf{x}_q), h_1(\mathbf{x}_i)) + d_H(h_2(\mathbf{x}_q), h_2(\mathbf{x}_i))$, where the first distance is the same for the items in the same bucket. The final items are an intersection of those within the range distance d in the full length, and those are stored in the buckets whose key values are within the range distance d_{key} from the query vector. The near duplicate image set is represented by $\mathcal{N}_h(\mathbf{x}_q) = \{\mathbf{x}_i | d_H(h_1(\mathbf{x}_q), h_1(\mathbf{x}_i)) < d_{key}\} \cap \{\mathbf{x}_i | d_H(h(\mathbf{x}_q), h(\mathbf{x}_i)) < d\}$, for all the database images I_i .

In this method, there are three important parameters to be determined: the hash key size l and the ranges d_{key} and d (or $d_{res} = d - d_{key}$). The hash key size l is related to the size of hash table, and it also limits the maximum recall of the near duplicates. If the size becomes larger, i.e., the hash table size increases, the achievable maximum recall increases. In other words, the larger values give better recall but need larger memory size and more buckets to be searched. The distance range d_{key} in the hash table determines the hamming ball size achieving the maximum recall or the best recall with small hamming ball size. The last parameter d is the range parameter, which determines the maximum detection rate for the allowed false alarm given by the system.

4 Multi-block Image Description

Although the Gist feature is effective in near-duplicate image representation, it is limited as a global feature. It is effective for encoding the holistic region, but the local image region or local details cannot be explicitly encoded. Therefore, the performance degradation is seen in strong variations, such as cropping and border framing. We solve the problem using a multi-block approach, which encodes a cropped region as well as the holistic region of an original image.

We use multiple query images instead of a single query image. The multiple query images include the original query image and its cropped images. Although other image variations can be further used, the cropping variation is most effective in our near-duplicate image detection scenario. We call it the multi-block image description because it is very similar to the multi-block approach used in face recognition [10].

The incorporation of the encoding of the cropped region plays several roles in near duplicate image detection. First, since the cropped image regions not only corresponds to one instant of cropping variations of a query image but also removes the border framing from the database images, it can deal with some of typical and error-prone variations of the near duplicate images. The cropped image region acts as a new query, and it is a kind of query expansion, which turns out to be very effective in bag-of-feature methods for local features. Second, a hybrid feature from the multiple image regions effectively increases the discriminability of the feature because the cropped region gives a slight different perceptual set of the Gist features, which results in more discriminability.

The multi-block approach produces multiple binary codes for each image, so they should be combined. Two alternative method is possible to compute distance. First, the multiple codes can be concatenated into a single binary code, but this approach increase the code length and it is hard to combine with the previous hash-based large scale search. Another approach is to multiple binary codes for each image and compare all possible distances and take the minimum distance. This is very natural in the context of hash-table-based search. Given a query image, the query

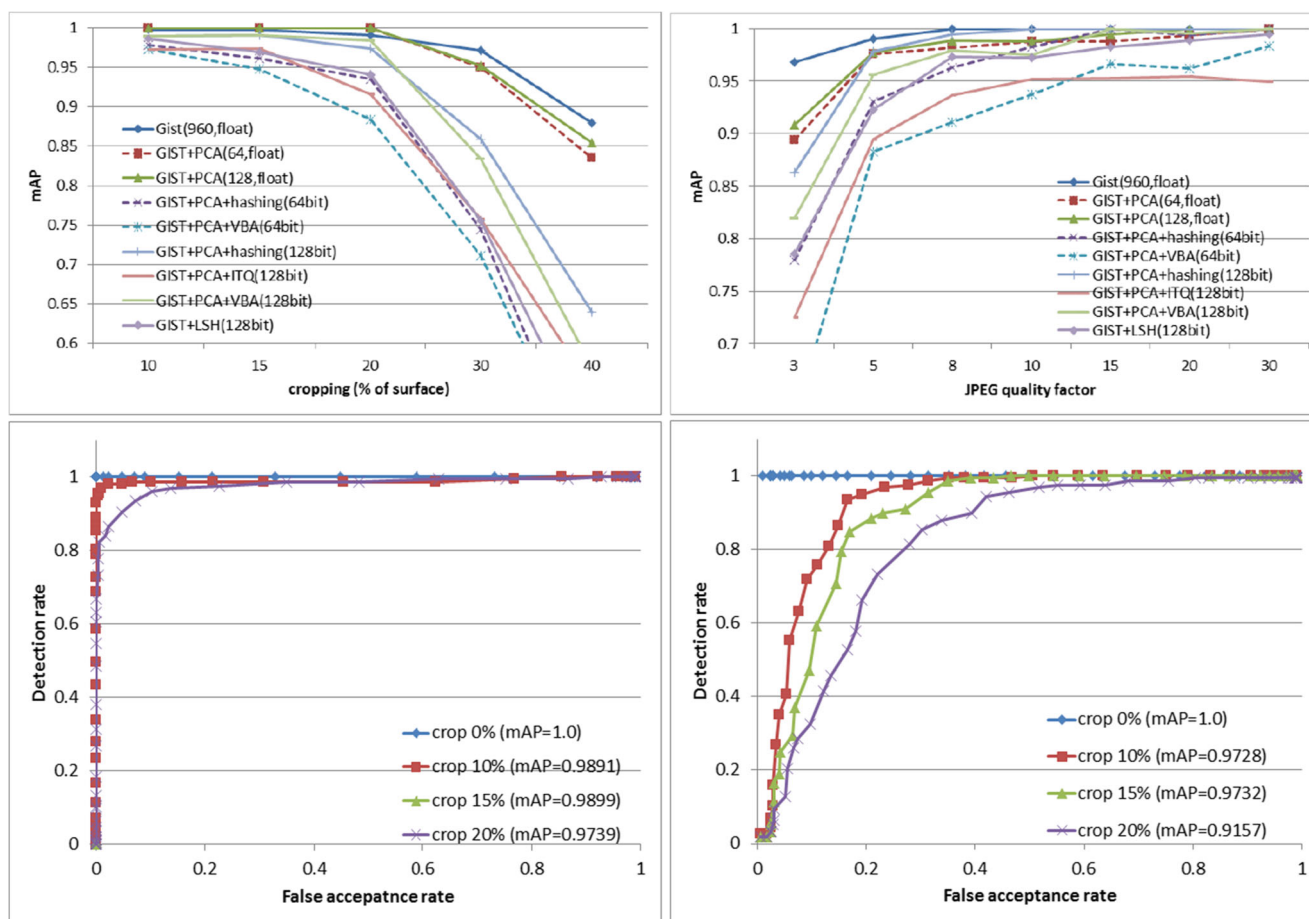


Figure 4 Comparison study. (Top row) mean average precision in terms of cropping and jpeg compression. (Bottom row) The ROC curve of the proposed code (Left) and the iterative quantization code.

image has multiple binary features. For each binary features, the possible candidates of the near-duplicate images have retrieved. Without directly computing minimum distances of the possible pairs, we decide the image as a near-duplicate image only if one of the binary code is shorter than the query.

5 Experimental Results

We experiment on publicly available datasets, including Copydays [7], Peekaboom [1], Mirflickr [9], and Tiny images [20]. Copydays are mostly used to test the accuracy of the compared methods, and Mirflickr and Tiny images act as distractors for scalability tests. For training the transform matrix and tuning the parameters, a Peekaboom dataset is used. Half of the set is used for training and the other half is used for tuning the parameters and for evaluation. For an accuracy evaluation, we measure the mean average precision (mAP), which is computed from a precision/recall curve, and we measure how many relevant images are searched in

the ranks [7]. Additionally, the detection rate (true positive) and false acceptance rate (false positive) are used to evaluate the performance as a detector or filter.

The near duplicate variations are tested in two ways. The variations provided in the Copydays dataset is first used, and they includes image croppings in terms of surface ratio and jpeg compressions with image scale reduction. In addition, more variations are generated to simulate the diverse near duplicate images observed in real image collections, as shown in Fig. 1.

5.1 More Comparison Study

We compare the proposed method (Gist+PCA+hashing) with the state-of-the-art methods: the Gist descriptor (Gist) and its PCA projection (Gist+PCA); the hashing methods (Gist + PCA + VBA, Gist+PCA+ITQ, GIST+LSH), which correspond to the optimal bit allocation [3]; iterative quantization [8]; and locality sensitive hashing by random projection. They are also compared in the different dimension

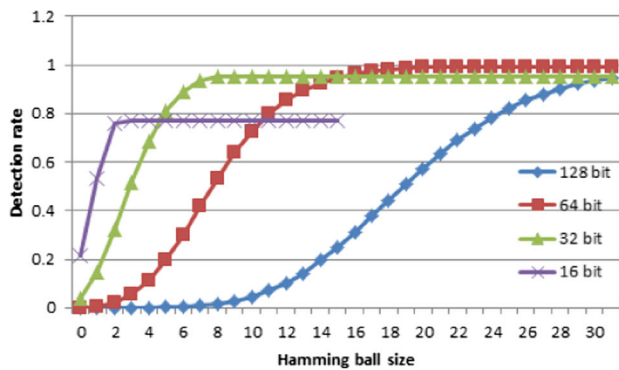
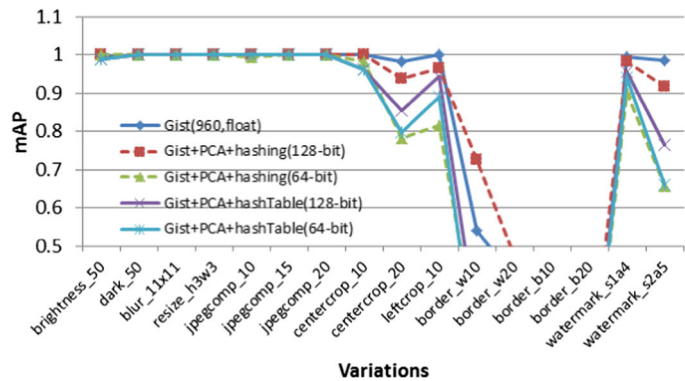


Figure 5 The hash table parameters and the decomposition verification. (Left) The detection rates plot with respect to hamming ball size for hash codes with various lengths. (Right) The compared mean average precision plots with respect to the near duplicate variations and



and code sizes: 64 and 128. The real-valued descriptors (Gist, Gist+PCA) are the benchmarks for the hashing techniques, and they are considered an upper bound for the performance.

The first rows of Fig. 4 showed comparison results for cropping and jpeg compression on the Copydays dataset in terms of mAP for 157 queries. The proposed method with 128-bits are the best performed in both cases, and the optimal bit allocation (Gist + PCA + VBA) follows. Interestingly, the 64-bit binary codes are comparable to other hashing techniques (Gist+PCA+ITQ, GIST+LSH) with 128-bit length. In the rows in Fig. 4, the ROC curves are presented to show the detection and false acceptance rates for the proposed code (Gist+PCA+hashing) and the iterative quantization code (Gist+PCA+ITQ). When the false acceptance rate is allowed at 0.05 (5%) in a system, the proposed method can achieve an almost 90% detection rate up to 20% cropping, while the iterative quantization method remains below 50%.

the hashing methods before and after the code decomposition. The decomposition code is represented by “Gist+PCA+hashTable” in the following plots because it uses the hash table for speedup.

5.2 Decomposition and Parameter Tuning

To decompose the full-length binary code, we need to find the hash key length and the hamming ball size for the maximum recall, i.e., l and d_{key} . We parameterize the hash key length with 16, 32, 64, and 128 bits for 128-bit binary code in full-length, and at each length, the detection rate is computed with respect to the hamming ball size, i.e., the distance threshold. We set the parameters at the 10% cropped images (centercrop_10), i.e., 19% cropping surface, which is a strong attack. As shown in Fig. 5(left), For 16, 32, 64, and 128 bits, the maximum detection rate is achieved up to approximately 77%, 95%, 97%, and 99% with a hamming ball size of 4, 9, 21, and 59, respectively. Therefore, we set the parameters: $l = 16$ and $d_{key} = 4$, and in this setting, only 77% of near duplicate images can be tested for further full-length comparison. This results in inevitable accuracy degradation for the sake of real-time detection. The final range threshold d is determined in the training stage when

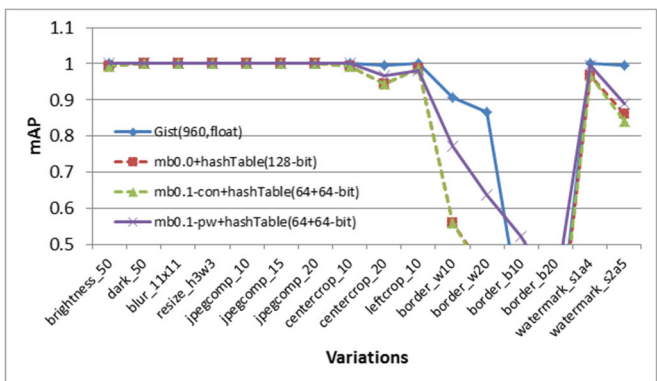
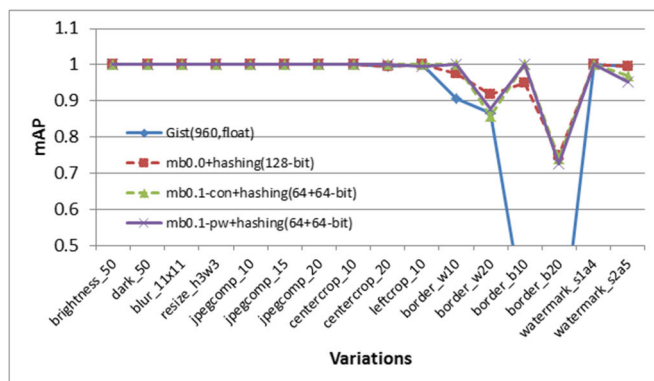


Figure 6 Multi-block coding. (Left) The full-length code. (Right) The decomposed code using the hash table. The number after “mb” is the cropping rate for the multi-block region, and mb0.0 means the

multi-block scheme is not applied. The following “con” and “pw” means the codes are compared by concatenating the vectors or a pairwise comparison between multiple blocks, respectively.

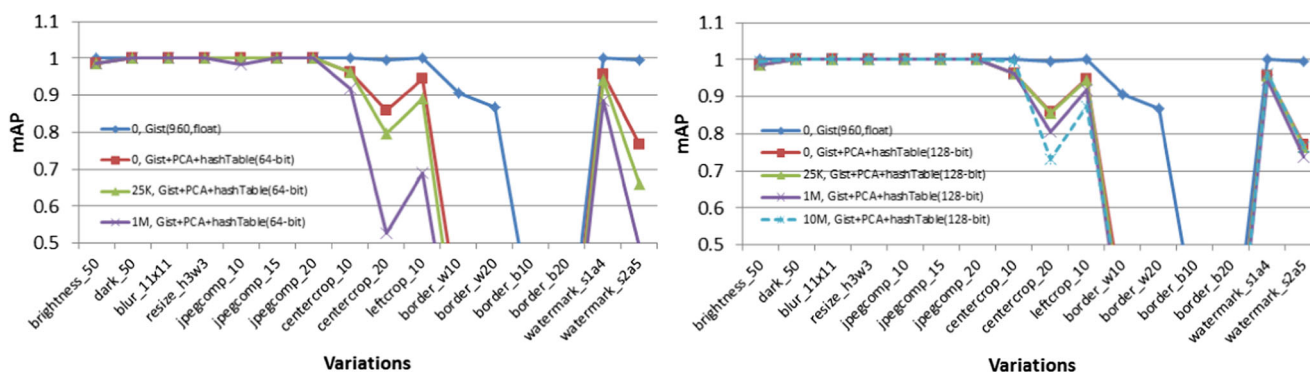


Figure 7 Scalability test. (Left) 64-bit code. (Right) 128-bit code. Note that multi-block approach is not applied in the experiments.

the allowed false acceptance rate is given. Typically, it ranges from 26 to 29 at 5% false acceptance.

The accuracy degradation by the decomposition is shown in Fig. 5. In this experiment, 25,000 images from Mirflickr are used as distractors, and Copydays are used for a test set. The degradation can be neglected up to centercrop 10% with 0.95 in mAP, and the mean average precision values are still acceptable for leftcrop 10% and centercrop 20% with 0.94 and 0.86, respectively. Even in the centercrop 20%, the degradation rate is less than 10%.

5.3 Effectiveness of the Multi-block Coding

The effectiveness of the multi-block encoding is shown in Fig. 6. When the full-length code is considered, the multi-block approach is slightly better than the PCA hashing, as was expected in some cases: border_w10 and border_b10. For the decomposed codes, the improvement is huge and the mean average precision remains high. For the border variation, the gain is over 2.1 and up to 3.1 in mean average precision. That shows that the multi-block approach is very effective for large-scale systems with strong variations.

5.4 Scalability to Large-Scale Images

Finally, scalability of the method is tested by increasing the size of the distractors: 0 (no distractor), 25,000 (Mirflickr 25K), 1,000,000 (Mirflickr 1M), and 10,000,000 (10 M random sample from Tiny images), Fig. 7 shows the results for both 64-bit and 128-bit binary codes. For 64-bit code, the mAP decreases very quickly and only centercrop 10% can be handled well. However, for 128-bit code, the mAP is good for the challenging variations, such as centercrop

20%, leftcrop 10%, and watermark_s2a5 under 10 million distractors.

To compute speed and memory usage at the larger scale, 29,000 queries from Peekaboom dataset for the centercrop 10% variation under 10 million distractors were tested and the results are averaged for stable computation. The RAM usage was 600M bytes, and the detection speed was 0.2 seconds on average on a single thread of 2.40GHz Intel Xeon CPU. The detection accuracy is 0.89 in mean average precision. When the hamming ball size of the hash table is reduced from three to four for the sake of accuracy, the speed is faster up to 0.07 sec, and the accuracy degradation is not big, 0.85 in mean average precision.

Finally, the algorithm is evaluated on the video data set with manual ground truth. We crawled 73,100 video thumbnails (i.e., 438,600 images, considering that each video has six thumbnails) with the most popular 673 keywords, and the near duplicate images are carefully verified by a human expert within each keywords. The different videos are determined as duplicates if more than three thumbnails are within a certain range. We set the false acceptance rate at 1.1%, and the detection rate of the algorithm is 78.4% with the distance range $d = 26$.

5.5 Computation Time

As shown in Table 2, PCAH are the most efficient method for training except for LSH, which does not require any training process. Supervised hashing methods such as MLH [18] and KSH [16] adopts optimization process for training, and requires much more time than other compared methods. Note that PCAH, ITQ, MLH and LSH spends same time for testing, because they just multiply the transform matrix

Table 2 Training and testing time of each method on CIFAR100 dataset [15] + 4k distractor.

	PCAH	KSH [16]	MLH [18]	ITQ [8]	SH [26]	LSH [11]
$T_{train,32bit}$	0.138	128	964	0.349	0.264	—
$T_{test,32bit}$	0.0234	0.267	0.0234	0.0234	0.105	0.0234

of the same dimension ($\mathbf{W}^T \in \mathbb{R}^{d \times d'}$). However, the test time of KSH and SH are much larger than others because KSH needs to compute the distance between anchor and test data, and SH uses multiple threshold for a single dimension. All the experiments were implemented in MATLAB and performed on the desktop with Intel-core i7 CPU at 3.30GHz and 16GB RAM.

6 Concluding Remarks

We proposed a scalable near duplicate image detection method by decomposing the binary code of Gist-PCA hashing and extending the holistic image region into multiple blocks. Comparison studies and experiment results verified the effectiveness in terms of accuracy, speed, and memory usage. In future, we will find a more compact code with sophisticated machine learning techniques and also extend this work to similar image search problem.

Acknowledgements This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) 2010-0028680 and 2014-003140.

References

1. von Ahn, L., Liu, R., Blum, M.: Peekaboom: a game for locating objects in images. In: Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI '06, pp. 55–64. ACM, New York, NY, USA (2006).
2. Baluja, S., Covell, M.: Beyond near duplicates : Learning hash codes for efficient similar-image retrieval. In: 20th International Conference on Pattern Recognition, pp. 543–547. IEEE (2010).
3. Brandt, J.: Transform coding for fast approximate nearest neighbor search in high dimensions. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13–18 June 2010, pp. 1815–1822. IEEE (2010).
4. Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S.S., Singh, J., Girod, B.: Transform coding of image feature descriptors. Proceedings of SPIE 7257, 725,710–725,710–9 (2009).
5. Chum, O., Philbin, J., Isard, M., Zisserman, A.: Scalable near identical image and shot detection. In: Proceedings of the ACM International Conference on Image and Video Retrieval (2007).
6. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: Proceedings of the British Machine Vision Conference (2008).
7. Douze, M., Jégou, H., Harsimrat, S., Amsaleg, L., Schmid, C.: Evaluation of GIST descriptors for web-scale image search. In: International Conference on Image and Video Retrieval. Santorini, Greece (2009).
8. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: CVPR, pp. 817–824 (2011).
9. Huiskes, M.J., Lew, M.S.: The mir flickr retrieval evaluation. In: MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval. ACM, New York, NY, USA (2008).
10. Hwang, W., Wang, H., Kim, H., Kee, S.C., Kim, J.: Face recognition system using multiple face model of hybrid fourier feature under uncontrolled illumination variation. IEEE Transactions on Image Processing 20(4), 1152–1165 (2011).
11. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pages 604–613. ACM, 1998.
12. Isard, M.: Bundling features for large scale partial-duplicate web image search. IEEE Conference on Computer Vision and Pattern Recognition (2009) pp. 25–32 (2009).
13. Jegou, H., Douze, M., Schmid, C., Prez, P.: Aggregating local descriptors into a compact image representation. In: CVPR, pp. 3304–3311. IEEE (2010).
14. Ke, Y., Sukthankar, R., Huston, L., Ke, Y., Sukthankar, R.: Efficient near-duplicate detection and sub-image retrieval. In: In ACM Multimedia, pp. 869–876 (2004).
15. Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Tech. Rep. 2009.
16. Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2074–2081. IEEE, 2012.
17. L. Liu, Y. Lu and C. Y. Suen, “Variable-Length Signature for Near-Duplicate Image Matching,” IEEE Transactions on Image Processing, vol. 24, no. 4, pp. 1282–1296, April 2015.
18. Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pages 353–360, 2011.
19. Oliva, A., Torralba, A.: Modeling the shape of the scene : A holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001).
20. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. 30(11), 1958–1970 (2008).
21. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large image databases for recognition. In: CVPR (2008).
22. Wang, B., Li, Z., Li, M., Ma, W.: Large-scale duplicate detection for web image search. In: IEEE International Conference on Multimedia and Expo, pp. 353–356 (2006).
23. Wang, J., Kumar, O., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, pp. 3424–3431 (2010).
24. Wang, X.J., Zhang, L., Liu, M., Li, Y., Ma, W.Y.: Arista - image search to annotation on billions of web photos. In: CVPR, p. 2987–2994 (2010).
25. Wang, Z., Josephson, W., Lv, Q., Charikar, M., Li, K.: Filtering image spam with near-duplicate detection. In: In Proceedings of the Fourth Conference on Email and AntiSpam, CEAS 2007 (2007).
26. Weiss, Y., Torralba, A.B., Fergus, R.: Spectral hashing. In: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (eds.) NIPS, pp. 1753–1760. MIT Press (2008).
27. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer (2006).
28. Zhang, L., Chen, L., Jing, F., Deng, K., Ma, W.Y.: Enjoyphoto: a vertical image search engine for enjoying high-quality photos. In: 14th Annual ACM International Conference on Multimedia, pp. 367–376 (2006).
29. L. Zheng, Y. Lei, G. Qiu and J. Huang, “Near-Duplicate Image Detection in a Visually Salient Riemannian Space,” IEEE Transactions on Information Forensics and Security, vol. 7, no. 5, pp. 1578–1593, Oct. 2012.



Hyunwoo Kim received a BS degree in Electronic Communication Engineering from Hanyang University, Seoul, Korea, in 1994, and MS and PhD degrees in Electrical and Computer Engineering from POSTECH, Pohang, Korea, in 1996 and 2001, respectively. In 2000 he was a visiting scientist in the Institute for Robotics and Intelligent Systems at University of Southern California, Los Angeles, CA. After getting a PhD degree, he worked at Samsung Electron-

ics Co., Ltd. (research staff), Korean German Institute of Technology (assistant professor), and Kakao Corp. (research scientist) in the field of image processing, computer vision, robotics, and human-computer interaction. In 2017, he joined Beijing Institute of Technology, Beijing, China, where he is currently an Associate Professor and a Principal Investigator of Advanced Innovation Centre of Intelligent Robots and Systems. His current research interests include computer vision, machine learning, artificial intelligence, and robotics.



Sungryull Sohn (S13) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2012 and 2013, respectively. From 2013 to 2016, he was with the Electronics and Telecommunications Research Institute, Korea, as a Researcher. He is currently pursuing Ph.D. degree in Computer Science Engineering from the University of Michigan, Ann Arbor, USA.

His research interests include machine learning, computer vision, and reinforcement learning.



Junmo Kim (S01-M05) received the B.S. degree from Seoul National University, Seoul, Korea, in 1998, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, in 2000 and 2005, respectively. From 2005 to 2009, he was with the Samsung Advanced Institute of Technology (SAIT), Korea, as a Research Staff Member. He joined the faculty of KAIST in 2009, where he is currently an Associate Professor of

electrical engineering. His research interests are in image processing, computer vision, statistical signal processing, machine learning, and information theory.