Supervised Multi-scale Locality Sensitive Hashing

Supervised Multi-scale Locality Sensitive Hashing

Li Weng [*]
LinkMedia group
Inria Rennes - Bretagne Atlantique
35042 Rennes, France

I-Hong Jhuo [†]
Institute of Information Science
Academia Sinica
11529 Taipei, Taiwan

Miaojing Shi
Key Laboratory of Machine Perception
Peking University
100871 Beijing, China

Meng Sun [‡]
IIP Lab, PLA University of Science and Technology
210007 Nanjing, China

Wen-Huang Cheng
MCLab, CITI
Academia Sinica
11529 Taipei, Taiwan

Laurent Amsaleg
IRISA Lab
CNRS Rennes
35042 Rennes, France

ABSTRACT

LSH is a popular framework to generate compact representations of multimedia data, which can be used for content based search. However, the performance of LSH is limited by its unsupervised nature and the underlying feature scale. In this work, we propose to improve LSH by incorporating two elements – supervised hash bit selection and multi-scale feature representation. First, a feature vector is represented by multiple scales. At each scale, the feature vector is divided into segments. The size of a segment is decreased gradually to make the representation correspond to a coarse-to-fine view of the feature. Then each segment is hashed to generate more bits than the target hash length. Finally the best ones are selected from the hash bit pool according to the notion of bit reliability, which is estimated by bit-level hypothesis testing.

Extensive experiments have been performed to validate the proposal in two applications: near-duplicate image detection and approximate feature distance estimation. We first demonstrate that the feature scale can influence performance, which is often a neglected factor. Then we show that the proposed supervision method is effective. In particular, the performance increases with the size of the hash bit pool. Finally, the two elements are put together. The integrated scheme exhibits further improved performance.

[*]L. Weng was supported by the French project Secular under grant ANR-12-CORD-0014.

[†]I-H. Jhuo is a co-first author. He and W.-H. Cheng were supported by the Ministry of Science and Technology of Taiwan under grant MOST-103-2911-I-001-531.

[‡]M. Sun was supported by the National Natural Science Foundation of China under grant 61402519 and the Natural Science Foundation of Jiangsu Province under grant BK20140071.

ICMR'15, June 23˜ 26, 2015, Shanghai, China.

http://dx.doi.org/10.1145/2671188.2749291.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.4.7 [**Computing Methodologies**]: Image Processing and Computer Vision—*feature representation*

General Terms

Algorithms, Design

Keywords

perceptual image hash, locality sensitive hashing, robust representation, multiple scale, supervised feature selection

1. INTRODUCTION

Hash algorithms for multimedia data have recently received much attention, because the compactness of hash values is the key for indexing and search in large-scale database systems. A hash value is typically a short binary string, whose length varies from tens to thousands of bits. It is a compact digest of the input data to a hash algorithm. In order to support content-based similarity search, multimedia hash algorithms emerged in recent years. They are typically designed to be *robust*, i.e., the hash value is independent of the binary representation of a multimedia object. On the other hand, they are also *discriminative*, i.e., different content should have different hash values.

In general, hashing techniques for multimedia data can divide into two categories – perceptual hashing and semantic hashing. They cover three applications – content classification, content identification, and content authentication. Existing algorithms generally differ in two aspects: 1) whether particular features are required; 2) whether training is required. Perceptual hashing [9] mainly deals with the latter two applications. Corresponding algorithms are typically feature-dependent, and do not require training. Semantic hashing [15], on the other hand, mainly addresses content classification. Corresponding algorithms are typically feature-independent and require training.

In this work, we focus on a class of feature-independent hash algorithms, called locality-sensitive hashing (LSH) [1]. LSH is a generic framework originally used for approximate nearest neighbor (ANN) search. An LSH scheme is a distribution on a family F of hash functions operating on a

collection of objects, such that for two objects x, y,

$$\Pr_{h\in F}[h(x) = h(y)] = sim(x, y), \quad (1)$$

where $sim(x, y) \in [0, 1]$ is some similarity function defined on the collection of objects, and Pr means probability. A popular implementation of LSH is based on scalar quantization [17]:

$$h_{r,b}(v) = \left\lfloor \frac{r \cdot v + b}{w} \right\rfloor, \quad (2)$$

where $\lfloor \cdot \rfloor$ is the floor operation, v is a feature vector, r is a random Gaussian vector, w is a quantization step, and b is a random variable uniformly distributed between 0 and w. In this work, our implementation of LSH is based on Charikar's work [3]:

$$h_r(v) = \begin{cases} 1 & \text{if } v \cdot r \geq 0 \\ 0 & \text{if } v \cdot r < 0 \end{cases} \quad (3)$$

This implementation actually measures the angular similarity between two feature vectors:

$$\Pr[h_r(u) = h_r(v)] \quad = \quad 1 - \frac{\theta(u, v)}{\pi}, \quad (4)$$

where $\theta(u, v) = \cos^{-1} \frac{u \cdot v}{||u|| \cdot ||v||}$ is the angle between u and v. This representation is the foundation of random projection based hash algorithms. In order to approximately quantize a feature vector, hyperplanes are randomly generated. The encoding depends on the relationship between the hyperplanes and the feature vector. The essential difference between LSH and later approaches lies in the way that hyperplanes are generated. Instead of using random hyperplanes, supervised algorithms try to search for hyperplanes that are more suitable for the problem at hand.

1.1 Contribution

In this paper, we propose an extension of LSH, which we call Supervised Multi-scale LSH (SMLSH). Two approaches are explored – supervised hash bit selection and multi-scale feature representation. Specifically, a feature vector is first represented by multiple scales; then each scale is hashed to generate more bits than the target hash length; finally, the best hash bits are selected from the candidate bit pool. This extension can effectively improve the performance of LSH in various applications with the following desirable properties:

- Compatibility to existing LSH schemes;
- Asymptotically guaranteed effectiveness;
- Scalability to large hash lengths.

The main advantage of the proposal is its versatility and thus the potential to be applied to other feature-independent hash algorithms. As an extension framework, we do not significantly modify an existing LSH scheme, so that conventional systems can be easily upgraded.

The scalability in hash lengths is very important for large-scale systems. According to the birthday paradox [18], one may find a pair of multimedia objects with the same n-bit hash value (a collision) among $2^{n/2}$ pairs. In practice, the collision rate can be higher for multimedia hashing due to the robustness requirement. Short hash lengths such as 32, 64 are more likely to cause false positives. In order to manage millions or billions of multimedia objects, a sufficient hash length is critical in a system design. A large hash length is also desirable for ANN applications where the conventional recall@R setting is used.

Existing supervised hash algorithms typically use various optimization techniques to compute hash bits. Due to the curse of dimensionality, this approach is intrinsically difficult when the hash length exceeds a certain level. SMLSH takes a different approach. It inherits the virtues of both supervised and randomized algorithms. As a randomized algorithm, SMLSH can easily extend to arbitrary hash lengths. As a *weakly* supervised approach, SMLSH does not greedily search for the best hyperplanes in order to be efficient and avoid over-fitting. Consequently, it improves performance with affordable complexity.

Multi-scale feature representation, to the best of our knowledge, is an unexplored approach in multimedia hashing. Existing algorithms typically assume a certain feature scale, which potentially limits performance. SMLSH unlocks this limit by considering multiple scales simultaneously.

1.2 Related work

Perceptual hashing started from the late 90's. Typical work includes Schneider and Chang's framework [16] and Fridrich's algorithm [5]. The latter is essentially a block-based LSH variant. Afterwards various algorithms based on different features are proposed, such as RASH [11] based on the Radon transform, Philips' audio hashing algorithm [7] based on the Mel-frequency cepstrum, the robust and secure hash based on the Fourier-Mellin transform [20]. Other features include higher-order statistics [25, 27], shapes [26], DFT phases [28], DCT or DWT signs [23, 24, 22], etc.

Feature-independent hashing or semantic hashing started from LSH. Typical work includes Charikar's LSH [3] for cosine similarity and Datar *et al.*'s LSH [4] for L_p distance. Later, various approaches are proposed to adapt the algorithm to the data and accommodate more semantics and modalities. For example, unsupervised training is used in spectral hashing [21], which is based on spectral clustering. The kernel trick is used in the Kernelized LSH [10]. Recently, supervised training is more widely used to overcome the semantic gap, such as [15, 6, 12].

2. SUPERVISED MULTI-SCALE LSH

Our goal is to improve LSH. Without loss of generality, the problem is defined as follows:

- Given an LSH algorithm with n-bit output, build a new algorithm with the same output length but improved performance.

In order to be versatile, we do not modify the internal realization of LSH. Since LSH can support arbitrary hash lengths, our solution to the above problem is the following:

- Given an LSH algorithm, generate n_o-bit output ($n_o \geq n$), form a hash value with improved performance by selecting n bits out of n_o bits.

The question is then how to select the bits. The key idea of SMLSH is that the choice of projections and features should both adapt to the problem. Thus SMLSH consists of two parts: multi-scale feature representation and hypothesis-testing-driven bit selection. A schematic diagram is shown in Fig. 1. The basic work-flow is the following:

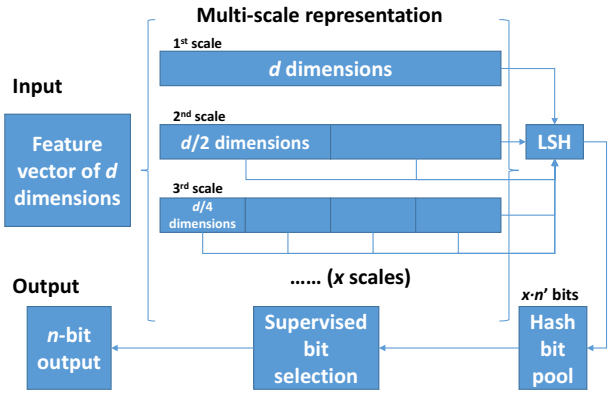


Figure 1: Schematic diagram of SMLSH.

1. The input feature vector is represented by x scales;
2. At each scale, the feature vector is fed into an LSH algorithm to obtain n' ($n' \geq n$) bits;
3. The best n bits are selected from all scales among the $n_o = x \cdot n'$ candidates.

In the following, the hash bit selection strategy and the multi-scale feature representation are described in detail.

2.1 Hash bit selection

Intuitively, we need to select the "best" n bits from the n_o-bit output. We realize it according to the criterion of *bit reliability*, a metric to measure the quality of each bit. It can be obtained through a training procedure. Once the bit reliability information is obtained, bit selection is just a sorting procedure:

1. Estimate the reliability of all n_o bits;
2. Sort the reliability of all n_o bits;
3. Output the most reliable n bits.

The above description gives an overview of the proposed scheme. Next, we define the bit reliability.

We consider an n-bit hash value as n binary classifiers, each represented by a single bit. The bit reliability can be evaluated by hypothesis testing. Denote the difference between two hash values at position i as $d_i \in \{0, 1\}$ ($i = 0, \cdots, n-1$). A decision is made from two hypotheses:

- $\mathbb{H}_0$ – the images correspond to irrelevant content;
- $\mathbb{H}_1$ – the images correspond to relevant content.

If $d_i = 0$, we choose $\mathbb{H}_1$; otherwise we choose $\mathbb{H}_0$.

The reliability of a hash bit can be characterized by the false positive rate p_{fp} and the false negative rate p_{fn}:

- p_{fp} = Probability $\{d_i = 0 | \mathbb{H}_0\}$;
- p_{fn} = Probability $\{d_i = 1 | \mathbb{H}_1\}$.

Overall, we define the bit reliability as

$$r_b = C_{fp} \cdot p_{fp} + C_{fn} \cdot p_{fn} \ , \tag{5}$$

where C_{fp} and C_{fn} are weight factors representing the cost for different mistakes. A smaller r_b corresponds to better reliability. This formulation is not biased by class skewness. It has some similarity to the objective function in LDA-Hash [19]. In the rest of the paper, we assume the weights are equal to 1/2.

If we obtain some ground truth labels for training, the bit reliability can be estimated. Thus we can improve an existing LSH scheme without modifying its internal realization.

2.2 Multi-scale feature representation

In practice, given a d-dimensional feature vector, an $\{l, k\}$ LSH scheme generates l sub-hash values, each with k bits. The two parameters l and k are important - the former typically corresponds to the number of hash tables; the latter is the size of a sub-hash value. The overall hash value consists of $l \times k$ bits. An interesting property of LSH is that it supports arbitrary hash lengths by varying l and k.

An often neglected factor in feature-independent hash algorithms is the *scale* of the feature vector. In order to hash (project) a feature vector, there are at least two ways: we could either compute $l \times k$ bits from the whole feature vector, or divide it into l parts and compute k hash bits from each part. Which approach is better?

This is similar to the question – whether we should use global or local features? In general, global features have good robustness but relatively weak discrimination, and local features show the opposite. For a certain problem, *one cannot decide in advance which scale is the best.* Therefore, we propose to test features of different scales and select the suitable ones.

Assume we consider x scales (Fig. 1). For each scale index $s_i = s_0 + i$ ($i = 0, 1, \cdots, x-1$), we evenly divide the feature vector into $l = 2^{s_i}$ parts and compute k_o ($k_o \geq k$) hash bits from each part, so that $n' = l \cdot k_o$. The parameter s_0 (set to 0 by default) decides the starting scale. The parameter x is determined in such a way that the minimum feature length $d/2^{s_0+x-1}$ is not too small. There are certainly other ways to construct feature vectors of different scales. We adopt our approach mainly because of the implementation simplicity.

3. PERFORMANCE AND COMPLEXITY

When the Hamming distance is used for hash comparison, two hash values are judged as relevant if their distance d is less than t. The performance of a hash algorithm can be characterized by the true positive rate P_{tp} and the false positive rate P_{fp}:

- P_{tp} = Probability $\{d < t | \mathbb{H}_1\}$;
- P_{fp} = Probability $\{d < t | \mathbb{H}_0\}$.

Assuming the n bits are independent and have average performance $\{p_{tp}, p_{fp}\}$, the performance of the overall scheme can be formulated as:

$$P_{tp} = f(p_{tp}) \tag{6}$$
$$P_{fp} = f(p_{fp}), \tag{7}$$

where $p_{tp} = 1 - p_{fn}$ and

$$f(p) = \sum_{k=n-t}^{n} \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}. \tag{8}$$

The above equation was used in Condorcet's jury theorem to show that a decision is more likely to be correct with more juries. In our proposal, the bit selection procedure essentially increases p_{tp} and decreases p_{fp} by replacing the original n bits with better candidates, i.e., we improve the quality of juries. Given a pool of n_o hyperplanes, the probability that our scheme fails is equal to the probability that

the original n bits are the best among the n_o choices, which is $1/\binom{n_o}{n}$. This probability can be made arbitrarily small by increasing n_o. In practice, this property asymptotically guarantees that our scheme is always effective. For example, $\binom{256}{128}$ is larger than 10^{15}.

Assuming each coefficient of a hyperplane is represented by b bit precision, for a feature vector segment of length d/l, there are totally $2^{d/l \cdot b}$ hyperplanes. That implies searching for a hyperplane in a high-dimensional space is computationally difficult. The training cost of greedy algorithms becomes prohibitively high for large hash lengths.

The computational cost of SMLSH consists of training cost and running cost. The training cost can be manually controlled. When there are N samples (e.g. images) available, there are maximum $\binom{N}{2}$ hash comparisons. We can have enough ground truths even when the training set is small. For example, $\binom{1000}{2}$ is approximately half a million.

The running cost depends on the implementation. In the worst case, when all the candidate hyperplanes are generated online by a pseudo-random number generator and are all used for projection (despite that not all results are used), the cost is about $x \cdot \frac{k_0}{k}$ times the cost of the original LSH. In practice, the computation can be reduced by pre-computing the selected hyperplanes offline.

4. EXPERIMENT

Since LSH is a general technique in content based search, we evaluate SMLSH in two different applications:

- Case 1: Near duplicate image detection;
- Case 2: Approximate feature distance estimation.

The former is related to content and copyright management; the latter is related to nearest neighbor search. The first application is a typical example with semantic gaps, i.e., relevant items do not necessarily result in small distances. The second application is an example of more ideal situations.

In the first application, SMLSH is used for identifying near-duplicate images. A near-duplicate is defined as a quasi-copy of an original multimedia object, typically resulted by incidental noise. We use 100 images to generate the training set and another 100 images to generate the testing set. They are randomly selected from the validation set of ILSVRC'2012.[1] Each set consists of $10,600$ images, including the 100 original ones and $10,500$ near-duplicates. The near-duplicates are created by applying a series of distortion to each of the 100 images. The list of distortion (15 categories, 7 levels each) is shown in Table 1. The relationship between the original images and their near-duplicates are used as the ground truth. GIST [14] feature vectors are extracted from all these images.

In the second application, SMLSH is used for estimating the similarity between SIFT vectors [13]. A dataset of ten thousand SIFT vectors is used [8].[2] Half of it is used for training and the other half is used for testing. The ground truth is set as follows: two SIFT vectors are determined as relevant if their cosine similarity is larger than 0.8.

Both datasets have been transformed by PCA in order to remove the correlation between feature dimensions. In particular, the GIST feature vectors are reduced to 256 dimensions. The SIFT vectors still keep 128 dimensions.

[1] `http://www.image-net.org/challenges/LSVRC/2012/`
[2] `http://corpus-texmex.irisa.fr`

Table 1: Distortions for near-duplicate generation.

Distortion name	Parameter range, step
1. Rotation	Angle: $1° - 7°$, $1°$
2. Central cropping	Percentage: $1\% - 7\%$, 1%
3. Row removal	Percentage: $1\% - 7\%$, 1%
4. Asymmetric cropping	Percentage: $1\% - 7\%$, 1%
5. Circular shifting	Percentage: $1\% - 7\%$, 1%
6. Down-scaling	Ratio: $0.7 - 0.1$, 0.1
7. Shearing	Percentage: $1\% - 7\%$, 1%
8. JPEG compression	Quality factor: $70 - 10$, 10
9. Median filter	Window size: $7 - 19$, 2
10. Gaussian filter	Window size: $7 - 19$, 2
11. Sharpening	Strength: $0.7 - 0.1$, 0.1[1]
12. Gaussian noise	PSNR: $45 - 15$ dB, 5 dB
13. Salt & pepper	Noise density: $0.01 - 0.07$, 0.01[2]
14. Gamma correction	Gamma: $0.5 - 1.7$, 0.2
15. Block tampering	Block number: $1 - 7$, 1[3]

[1] Parameters for the MATLAB function *fspecial('unsharp')*.
[2] Parameters for the MATLAB function *imnoise()*.
[3] The size of a block is 1/64 of an image.

Table 2: Notations of SMLSH.

Notation	Definition
n	Hash length (bits)
x	Number of scales
s_i	Scale index ($i = 0, \cdots x-1$)
k	Initial sub-hash size (for s_0)
l	Initial number of feature segments (for s_0)

4.1 Baselines and experiment setting

We mainly use the basic LSH algorithm defined in (3) as the baseline. Specifically, the first scale is used without supervision ($l = 1, k_o = k$). Another algorithm for performance comparison is the recently proposed qoLSH [2] in symmetric mode. It is only used in Fig. 3b and Fig. 5c for Case 2 to generate 256-bit hash values, because it requires the hash length to be larger than the number of feature dimensions while we have only tested hash lengths of 64, 128, and 256 so far. The experiments investigate the relationship between the performance and typically the following factors:

- The hash size (64, 128, 256);
- The size of the bit selection pool (200%, 400%);
- The number of available feature scales (1–3);

Hypothesis testing is used for evaluating SMLSH in both scenarios. The receiver operating characteristic (ROC) curves are used for representing the performance. The two cases take $\binom{10600}{2} = 56,174,700$ and $\binom{5000}{2} = 12,497,500$ pairwise comparisons respectively. We do not use a retrieval setting because we focus on the hash performance only.

In the following, we first evaluate the effects of single scales and supervision separately, then put them together. The notations used in this work are summarized in Table 2.

4.2 Effect of single feature scales

Recall that different feature scales may have different impacts on the performance. We consider the parameter settings listed in Table 3. For the same hash length n, different $\{k, l\}$ combinations are considered. In general, longer feature vectors are likely to enable more combinations.

The ROC curves are shown in Fig. 2 for the two scenarios. Results indicate that the feature scale indeed matters. In Case 1, when the false positive rate is small, the scale can make a big difference. In Case 2, the scale effect is not as

Table 3: Parameters for different feature scales.

n	64		128			256			
k	64	32	128	64	32	256	128	64	32
l	1	2	1	2	4	1	2	4	8
x	1								

Table 4: Parameters for different hash pool sizes.

n	64			128			256		
k	64	128	256	128	256	512	256	512	1024
l	1								

obvious as in Case 1 – the performance is not very different for the same hash length, but still there is always a small difference. In Case 1, the scale effect becomes more obvious with larger hash sizes, where a smaller scale is possible. On the other hand, a fine scale is not always better than a coarse scale. For example, the performance of the first two scales is not significantly different. That implies the scale effect is more important for larger hash sizes and thus for high dimensional feature vectors.

4.3 Effect of supervised bit selection

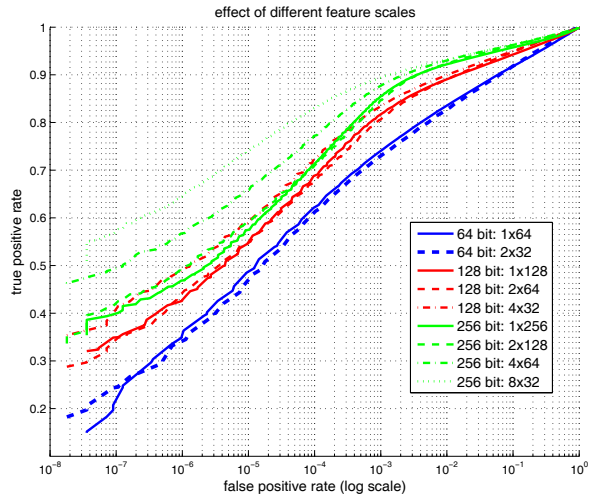
Next, we consider the effect of supervised bit selection. For the same hash length, different hash bit pool sizes are used. Only the first scale is used. The parameters are listed in Table 4.

The ROC curves are shown in Fig. 3 for the two scenarios. The results confirm that our proposed supervision approach is effective. The performance gain is significant for both cases, and increases with the hash bit pool size. This is consistent with intuition, because a larger bit pool is likely to provide better hyperplanes. For example, with proper supervision, 128 bits can even outperform 256 bits. In addition, Fig. 3b shows that qoLSH outperforms LSH at 256 bits, but supervised LSH performs even better.

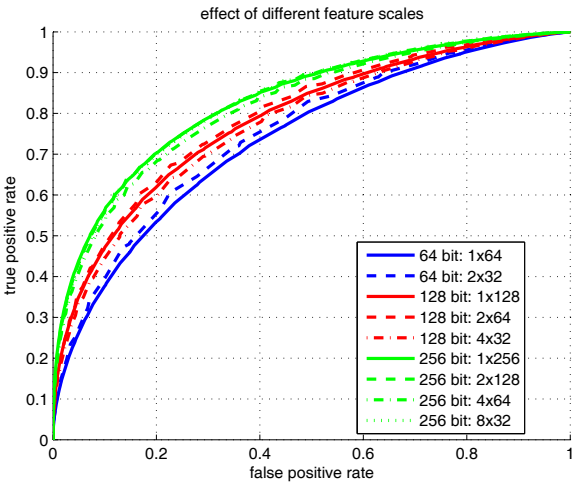
4.4 Put things together

Previously, we have observed that the performance increases with the size of the hash bit pool. In this section, we put together our two leverages, i.e., supervision based on multiple scales. The main parameters are listed in Table 5.

Figure 4 demonstrates the selection of 256 bits from four scales according to the bit reliability. The figure shows that each scale contributes a significant amount of bits. The ROC curves are shown in Fig. 5 for the two scenarios. The results show that supervision based on multiple scales can achieve even better performance. For example, "64/256,3" outperforms "64/256,1". In general, the performance increases with the number of scales, especially when there are more than two scales. It is clear for Case 2 (Fig. 5c-d), while Case 1 is worth more analysis. In Fig. 5a-b, when the number of scales increases from one to two, the ROC curves typically intersect at a certain middle point: on the left the performance is decreased and on the right the performance is increased. In other words, the true positive rate rises for high false positive rates and drops for low false positive rates. We call this phenomenon the "s-shape effect". Since the x-axis is in log-scale, although it is not very obvious in the figures, the overall performance is still increased. For the same scale number, the intersection point moves towards the right side (higher true positive rate) as we increase the hash length. When the scale number is further increased, the performance on the left region rises again and the intersection disappears.



(a) Case 1, GIST



(b) Case 2, SIFT

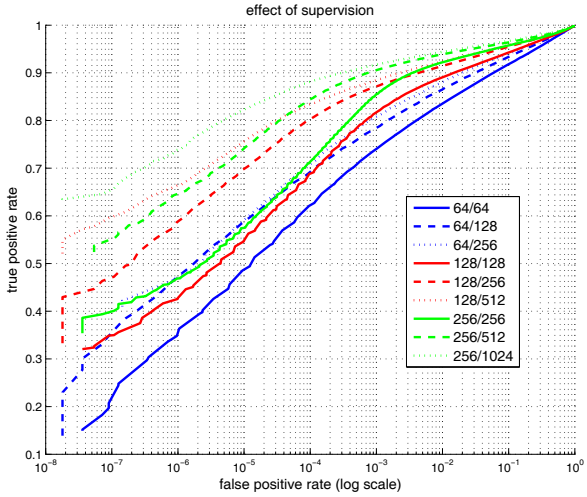
Figure 2: The effect of different scales. "n bit: $l \times k$": feature vector divided into l parts, k bits each. Different scales influence the performance.

Note that there are also some intersections at the left ends of the curves, but we are not really interested in that region due to lack of sufficient statistics. Additionally, Fig. 4c confirms again that SMLSH outperforms qoLSH at 256 bits.

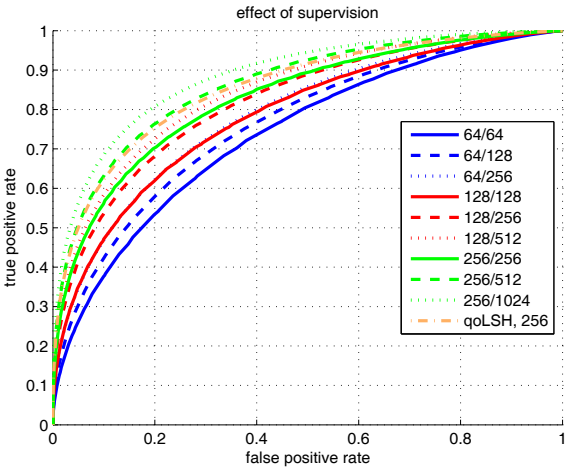
Figure 6 shows how the performance evolves with different configurations for 64 bits. It is interesting that for the same hash length the performance could vary so much. Therefore the effectiveness of SMLSH is verified.

5. DISCUSSION

Compared with other multimedia hash algorithms, the advantage of our proposal is that it improves performance through supervision, yet it is still a randomized algorithm. The complexity of the algorithm increases linearly with the hash length, which is a desirable property for large-scale multimedia data applications. The proposed concept of supervision based on multiple scales is not limited to LSH. In fact it applies to any hash algorithm that is able to generate sufficiently many bits, i.e., hyperplanes. In particular,



(a) Case 1, GIST



(b) Case 2, SIFT

Figure 3: The effect of supervision. "n/n_o": n bits selected from n_o bits. The performance increases with the hash bit pool size n_o. qoLSH 256 outperforms LSH 256 in (b).

SMLSH is not explicitly limited by feature dimensions, unlike e.g. qoLSH which is only effective when the hash length is larger than the number of feature dimensions.

An interesting question is what is the best number of scales for a particular application. Our experiment results show that generally it is better to have more than two scales. On the other hand, the number of scales cannot increase arbitrarily, because after a certain scale the feature is no longer meaningful and become unstable. In our near-duplicate detection example, we have observed the s-shape effect. This is essentially due to the semantic gap. Note that increasing the hash length on the same scale cannot avoid this effect, but increasing the number of scales can. From this point of view, SMLSH provides more flexible trade-offs between robustness and discrimination.

Analogous to approaches in chemistry and physics, the proposed method actually introduces a way to observe features in a *microscopic* way. We no longer look at features as a whole, no matter global or local. Instead we turn features into bits: those from large scales are like molecules, and those from small scales are like atoms; each of them carry different information, and can be utilized separately.

Table 5: Parameters for multi-scale supervision.

n	64			128			256		
k	128,256			256,512			512,1024		
l	1								
x	1	2	3	1	2	3	1	2	3

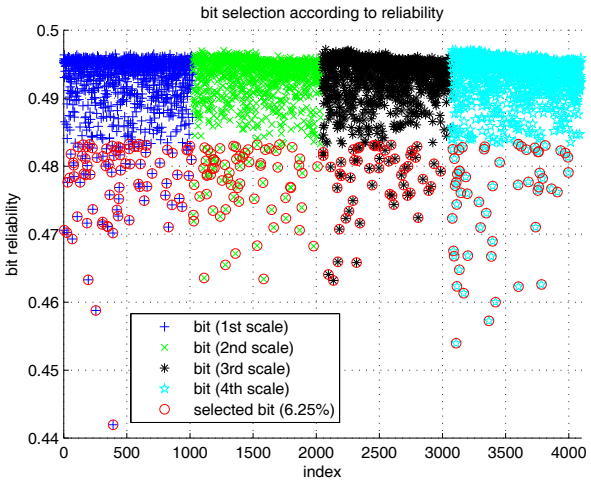


Figure 4: Bit selection from four scales. A small value means better reliability. Each scale contributes significantly.

Our method of constructing a multi-scale representation is quite simple. There are certainly other ways. For example, if some prior information about the feature is available, such as inherent structures, we may derive a better multi-scale representation. In our experiment, the GIST and SIFT features are relatively short compared with current high dimensional descriptors. With thousands of or even more feature dimensions, we conjecture that the scale effect can make significant influence on the performance.

6. CONCLUSION

In this work, we improve the classic LSH framework by incorporating two novel elements: supervised hash bit selection and multi-scale feature representation. The basic idea is to generate more bits than the target hash length, and select the best ones out of the hash bit pool. Each bit is considered as a weak binary classifier. The notion of bit reliability is used for estimating the quality of each bit, which is defined as a weighted average of the false positive rate and the false negative rate in a hypothesis test during the training stage. In addition, a feature vector is represented by multiple scales. At each scale, the feature vector is divided into segments, and each segment is independently hashed. The size of the segment is decreased gradually so that the hash bit pool corresponds to a coarse-to-fine view of the feature.

Extensive experiments have been performed based on hypothesis testing. Two applications are simulated: near-duplicate image detection and approximate feature distance estimation. We first demonstrate that the choice of feature scale indeed can make a difference in performance, which is an often neglected factor in content-based applications utilizing various sorts of features. Then we show that the proposed supervision method is effective. In particular, the

performance increases with the size of the hash bit pool. This is an intuitive result because our supervision approach is asymptotically guaranteed to be effective. Finally, the two elements are combined to obtain better performance. In addition, initial results imply that SMLSH outperforms the recently proposed qoLSH in a symmetric setting.

7. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. of 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006.

[2] R. Balu, T. Furon, and H. Jégou. Beyond "project and sign" for cosine estimation with binary codes. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6884–6888, May 2014.

[3] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of 34th ACM Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of 20th Symposium on Computational Geometry (SCG)*, pages 253–262, 2004.

[5] J. Fridrich. Robust bit extraction from images. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 536–540, 1999.

[6] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–824, June 2011.

[7] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proc. of 3rd International Conference on Music Information Retrieval*, pages 107–115, October 2002.

[8] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[9] F. Khelifi and J. Jiang. Perceptual image hashing based on virtual watermark detection. *IEEE Transactions on Image Processing*, 19(4):981–994, April 2010.

[10] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, June 2012.

[11] F. Lefèbvre, B. Macq, and J.-D. Legat. RASH: RAdon Soft Hash algorithm. In *Proc. of 11th European Signal Processing Conference*, volume 1, pages 299–302, Toulouse, France, Sep. 2002.

[12] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081, 2012.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

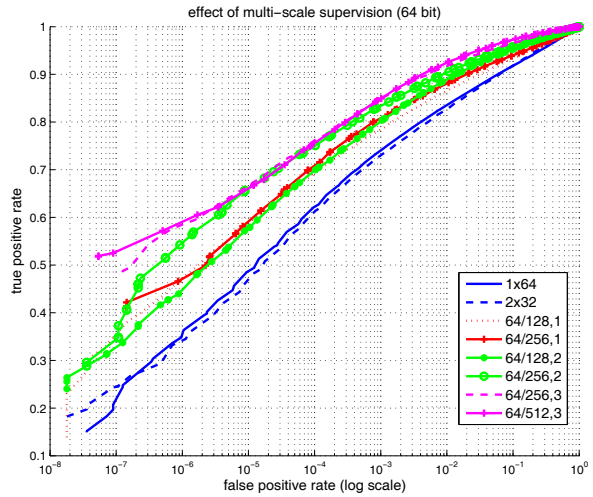
[14] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[15] R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proc. of International Conference on Artificial Intelligence and Statistics*, volume 11, pages 412–419, 2007.

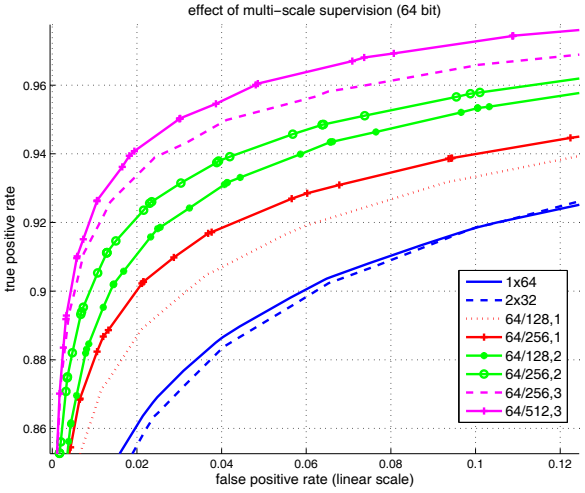
[16] M. Schneider and S.-F. Chang. A robust content based digital signature for image authentication. In *Proc. of International Conference on Image Processing (ICIP)*, volume 3, pages 227–230, 1996.

[17] M. Slaney and M. Casey. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *Signal Processing Magazine, IEEE*, 25(2):128–131, 2008.

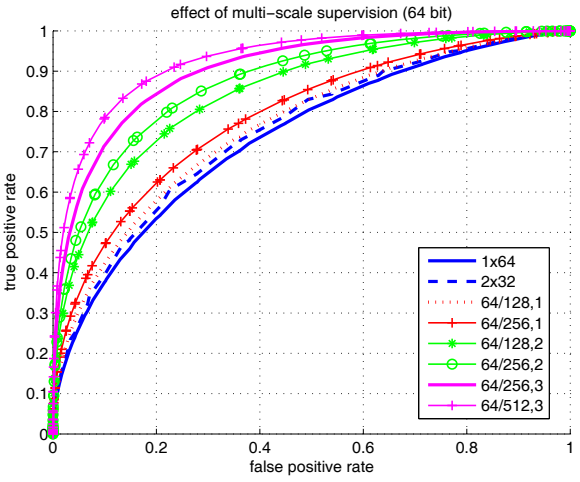
[18] W. Stallings. *Cryptography and Network Security*. Prentice Hall, 4th edition, 2005.



(a) Case 1, GIST

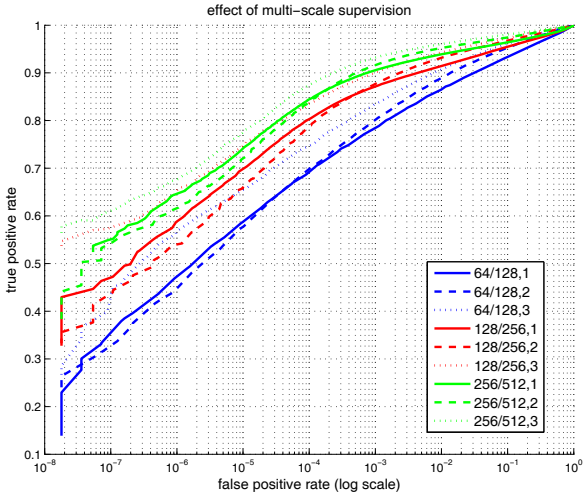


(b) Case 1, GIST, close-up

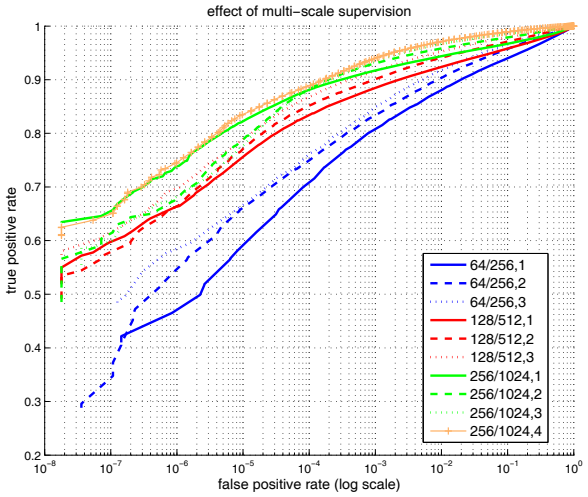


(c) Case 2, SIFT

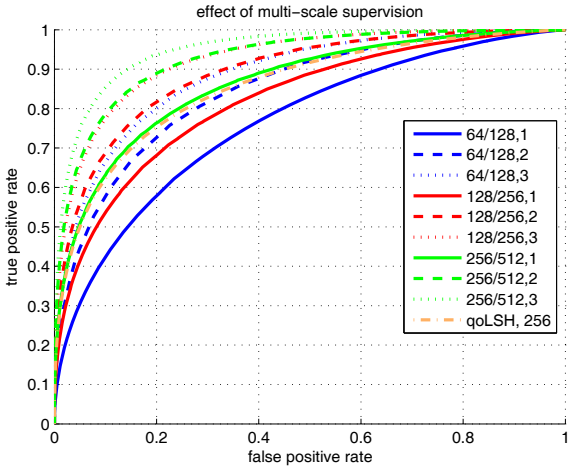
Figure 6: Different ways to generate the same hash length (64 bits) lead to different performance. "$n/n', x$": n bits selected from $n' \cdot x$ bits in x scales.



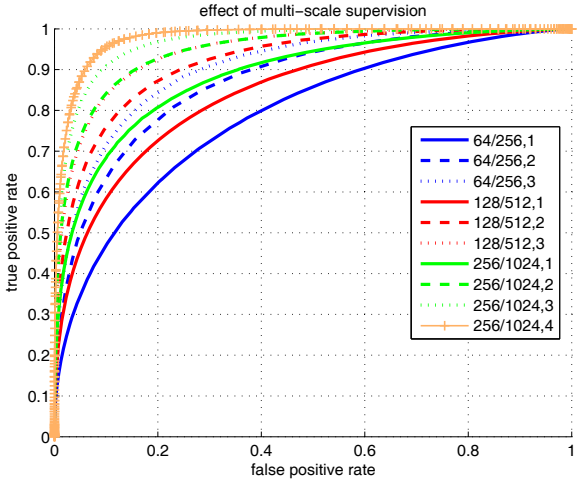
(a) Case 1, GIST, 200% hash bit pool size



(b) Case 1, GIST, 400% hash bit pool size



(c) Case 2, SIFT, 200% hash bit pool size



(d) Case 2, SIFT, 400% hash bit pool size

Figure 5: The effect of multi-scale supervision. "$n/n', x$": n bits selected from $n' \cdot x$ bits in x scales. The combined scheme can further improve the performance. SMLSH 256/512,1 outperforms qoLSH 256 in (c).

[19] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):66–78, 2012.

[20] A. Swaminathan, Y. Mao, and M. Wu. Robust and secure image hashing. *IEEE Transactions on Information Forensics and Security*, 1(2):215–230, June 2006.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[22] L. Weng, L. Amsaleg, A. Morton, and S. Marchand-Maillet. A privacy-preserving framework for large-scale content-based information retrieval. *IEEE Transactions on Information Forensics and Security*, 10(1):152–167, Jan. 2015.

[23] L. Weng, G. Braeckman, A. Dooms, and B. Preneel. Robust image content authentication with tamper location. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 380–385, 2012.

[24] L. Weng, R. Darazi, B. Preneel, B. Macq, and A. Dooms. Robust image content authentication using perceptual hashing and watermarking. In *Proc. of 13th Pacific-Rim Conference on Multimedia (PCM)*, volume 7674 of *LNCS*, pages 315–326, 2012.

[25] L. Weng and B. Preneel. On secure image hashing by higher-order statistics. In *Proc. of IEEE International Conference on Signal Processing and Communications*, pages 1063–1066, 2007.

[26] L. Weng and B. Preneel. Shape-based features for image hashing. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, pages 1074–1077, 2009.

[27] L. Weng and B. Preneel. A novel video hash algorithm. In *Proc. of ACM International Conference on Multimedia*, pages 739–742, October 2010.

[28] L. Weng and B. Preneel. A secure perceptual hash algorithm for image content authentication. In *Proc. of International Conference on Communications and Multimedia Security*, volume 7025 of *LNCS*, pages 108–121, 2011.