2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

A Feature Fusion Framework for Hashing

I-Hong Jhuo[1], Li Weng[2]* ✉, Wen-Huang Cheng[3], D. T. Lee[4]
[1]Columbia University, New York, U.S.A.
[2]IGN - French Mapping Agency, LaSTIG/MATIS Lab, 94165 Saint-Mandé, France
[3]MCLab, CITI, Academia Sinica, Taipei, Taiwan
[4]Institute of Information Science, Academia Sinica, Taipei, Taiwan
*(co-first author)

***Abstract*—A hash algorithm converts data into compact strings. In the multimedia domain, effective hashing is the key to large-scale similarity search in high-dimensional feature space. A limit of existing hashing techniques is that they typically use single features. In order to improve search performance, it is necessary to utilize multiple features. Due to the compactness requirement, concatenation of hash values from different features is not an optimal solution. Thus a fusion process is desired. In this paper, we solve the multiple feature fusion problem by a hash bit selection framework. Given multiple features, we derive an n-bit hash value of improved performance compared with hash values of the same length computed from each individual feature. The framework utilizes a feature-independent hash algorithm to generate a sufficient number of bits from each feature, and selects n bits from the hash bit pool by leveraging pair-wise label information. The metric bit reliability is used for ranking the bits. It is estimated by bit-level hypothesis testing. In addition, we also take into account the dependence among bits. A weighted graph is constructed for refined bit selection, where the bit reliability is used as vertex weights and the mutual information among hash bits is used as edge weights. We demonstrate our framework with LSH. Extensive experiments confirm that our method is effective, and outperforms several state-of-the-art methods.**

I. Introduction

In the big data era, the rapidly growing amount of text, audio and visual data on Internet is urging more effective ways of exploration. Among related issues, an important question is how to efficiently search for multimedia content in large-scale systems. Thus the similarity search problem has attracted increasing attention in data analysis or multimedia research community. Content-based search typically utilizes high-dimensional descriptors, such as SIFT [16], GIST [17], VLAD [9], MFCC [4], or learnt features [10]. These descriptors are effective, but their dimensionality is relatively large. They are not suitable for in-memory search when the database scale exceeds a certain level. In order to trade for efficiency, more compact descriptors are needed. A promising solution is *multimedia hashing*, which means to represent multimedia objects as compact hash values. It is essentially an embedding from a feature space to a Hamming space. A hash value is typically an n-bit string, whose compact size can facilitate in-memory indexing and search. Hash comparison can be efficiently achieved by the XOR operation. Being different from other hash algorithms, multimedia hash (MH) algorithms are *robust*, i.e., relevant content should result in similar hash values. On the other hand, they are also *discriminative*, i.e., different content should result in different hash values.

This work was partly supported by Ministry of Science and Technology under grants MOST 103-2911-I-001-531 and MOST 104-2911-I-001-506.

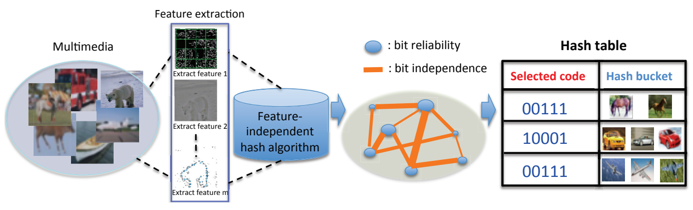

Fig. 1: The proposed feature fusion framework. Each data point is represented by multiple features. A hash bit pool is generated by applying a feature-independent MH algorithm to each feature. A refinement graph is used to select good bits based on reliability and independence.

In general, MH algorithms can be categorized as feature-dependent and feature-independent [27]. The former integrates with particular features, thus emphasizes feature extraction; the latter applies to any feature in vector form, typically focusing on feature transformation (learning) [23]. Depending on the purposes, some MH algorithms are used for identifying *perceptual* similarities, while some others are for *semantic* similarities.

A challenge to multimedia hashing is to utilize multiple features while maintaining hash compactness. An MH scheme is essentially a classifier based on some feature(s). Due to the compactness requirement and implementation complexity, existing MH algorithms typically use single features, such as shape features [28], pixel statistics [29], DFT coefficients [30], DCT/DWT coefficients [25], [26], etc. According to the *no-free-lunch* theorem [31], there is no single best classifier. In order to boost search performance, it is necessary to combine multiple features [11]. For example, a naive way is to hash each feature separately and concatenate the results. Another way is to concatenate the features and hash them together. They demonstrate different trade-offs between compactness and performance. On the one hand, compactness is the key for efficiency – the speed of hash comparison and the size of hash tables both depend on the hash length. On the other hand, compactness also limits the ultimate performance. Intuitively, an n-bit hash value can at most distinguish 2^n

items. In practice, n is typically less than a thousand. In some applications with real-time requirements, n is even less than 32 [21], which is not large enough to ensure low collision rates in practice. Given a feature set and an arbitrary hash length n, it is not straight-forward to generate optimal hash values. The problem is further complicated with multiple features. Ideally, we hope to benefit from more features without increasing n.

In this work, we propose a framework to solve the following *feature fusion* problem:

> Given m feature vectors $\mathbf{F}_i$ $(i = 0, 1, \cdots, m-1)$ of the same multimedia object, generate an n-bit hash value, which outperforms n-bit hash values generated from any single $\mathbf{F}_i$,

where $\mathbf{F}_i = [f_0, f_1, \cdots, f_{d_i}]^T$ has d_i dimensions. Our goal is to derive a generic solution that can be adapted to many MH algorithms, so that multiple features can be easily incorporated under constrained hash lengths. We use a feature-independent MH algorithm to generate hash bits from each feature. These bits are put into a hash bit pool. Finally, n bits are selected from the pool. A novel metric is used for evaluating the quality of hash bits. It is simple to compute because it only requires pair-wise label information. In addition, the dependence among hash bits is modeled as a graph. Effective and independent bits are selected by sub-graph extraction. Compared with other methods, our framework exhibits superior performance and flexibility.

II. Related work

Exploiting multiple features is related to feature pooling [3], fusion [6], or selection [7]. Sometimes these terms are interchangeably used in literature.

Strictly speaking, our proposal is a hash fusion method (different from general feature fusion), since we need a hash algorithm to convert features into bits. There is not much work on hash fusion. Some recent work considers hash fusion as a bit selection problem. Liu *et al.* model hash bit selection as a sub-graph selection problem [15], [33]. Hash bits are represented as vertices in a graph representation. The weight of a vertex is defined by its similarity preservation capability [24]; the weight of an edge is defined by the independence between the two vertex bits. A sub-graph with heavy weights is selected. A common challenge to graph-based approaches is the construction of large-scale graphs. When the number of hash bits is large, deriving relationship among hash bits can be time consuming. These approaches belong to late fusion.

In fact, many dimension reduction techniques can be used for feature fusion, such as principle component analysis (PCA) or graph embedding [32]. However, they typically need further processing, e.g., a quantization stage, to compute hash values. On the other hand, there are feature-independent hash algorithms, such as locality-sensitive hashing (LSH) [1]. Technically, a simple fusion method is to feed them with the concatenation of multiple feature vectors, although the performance is typically not the best. There are also hash algorithms that support multiple features as input, such as multiple feature hashing (MFH) [20] and multiple feature kernel hashing (MFKH) [14]. However, they are not flexible when the number of features is large, because the feature data has to be present at the same time for computing. These approaches belong to early fusion.

Our proposal is a novel bit selection method. It simultaneously considers semantic information and bit independence. Compared with existing late fusion approaches [15], [33], we use a novel metric bit reliability to estimate the importance of each bit. This metric is different from the similarity preservation [15], as it takes into account both robustness and discrimination.

III. The proposed feature fusion framework

The key idea of the proposed framework is to utilize feature-independent MH algorithms, as illustrated by Fig. 1. The fusion procedure consists of the following steps:

1) Each feature vector $\mathbf{F}_i$ is converted to a hash value of $n_i \geq n$ bits by a feature-independent MH algorithm;
2) All hash values are put into a hash bit pool of $n' = \sum n_i$ bits;
3) An n-bit hash value is created by selecting n bits from the hash bit pool.

This framework is flexible enough to adapt to different features and MH algorithms as long as they are feature-independent. More details are explained in the following.

A. The hash algorithm

We first need to adopt a feature-independent MH algorithm. A good candidate is locality-sensitive hashing (LSH) [1]. It is a generic framework for approximate nearest neighbor (ANN) search. An LSH scheme is a distribution on a family H of hash functions operating on a collection of objects, such that for two objects $\mathbf{x}$, $\mathbf{y}$,

$$\text{Pr}_{h \in H}[h(\mathbf{x}) = h(\mathbf{y})] = \text{sim}(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\text{sim}(\mathbf{x}, \mathbf{y}) \in [0, 1]$ is some similarity function defined on the collection of objects, and Pr means probability. A popular implementation of LSH is based on scalar quantization [19]. In this work, our implementation of LSH is based on Charikar's work [5]:

$$h_{\mathbf{r}}(\mathbf{v}) = \begin{cases} 1 & \text{if } \mathbf{v} \cdot \mathbf{r} \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $\mathbf{v}$ is a feature vector, $\mathbf{r}$ is a random Gaussian vector. This implementation actually measures the angular similarity between two feature vectors:

$$\text{Pr}[h_{\mathbf{r}}(\mathbf{u}) = h_{\mathbf{r}}(\mathbf{v})] \quad = \quad 1 - \frac{\theta(\mathbf{u}, \mathbf{v})}{\pi}, \tag{3}$$

$\theta(\mathbf{u}, \mathbf{v}) = \cos^{-1} \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}$ being the angle between $\mathbf{u}$ and $\mathbf{v}$.

After the MH algorithm is chosen, n_i hash bits should be generated from each single feature vector $\mathbf{F}_i$. An advantage of LSH is that an arbitrary number of bits can be generated by adjusting the number of random projections. Thus the hash bit pool can be made arbitrarily large.

B. Bit selection based on reliability

The most important part of the framework is the selection of hash bits. In this paper, we select best n bits according to a quality metric. There might be different metrics to measure the quality of hash bits, such as similarity preservation [24]. We find that a more suitable candidate is the recently proposed *bit reliability* [22], [27]. Denoted by r_b, it is defined as a weighted average of the false positive rate and the false negative rate of *each hash bit*. The concept of bit reliability was originally proposed by Voloshynovskiy et al. [22] from a channel coding perspective. It was generalized in [27] and shown to be practical when evaluating hash bits from a single feature. In this work, we find that this metric is also suitable for multiple feature fusion.

Following the definition in [27], we consider an n-bit hash value as n binary classifiers, each represented by a single bit. The bit reliability can be evaluated by a hypothesis test. Denote the difference between two hash values at position i as $e_i = \{0, 1\}$ $(i = 0, \cdots, n-1)$. When two hash values are compared, a decision is made from two hypotheses:

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

If $e_i = 0$, $\mathbb{H}_1$ is chosen; otherwise $\mathbb{H}_0$ is chosen. The bit reliability is characterized by the false positive rate p_{fp} and the false negative rate p_{fn} of a hash bit:

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

which measure discrimination and robustness respectively. Overall, the bit reliability is defined as:

$$r_b = C_{fp} \cdot p_{fp} + C_{fn} \cdot p_{fn} \ , \tag{4}$$

where C_{fp} and C_{fn} are weight factors (set to 0.5 in our experiments). A *smaller* r_b corresponds to better reliability. After the reliability is computed, bit selection becomes a sorting procedure. In practice, the weights can be adjusted according to the application. For example, biometric applications typically require small false positive rates, i.e. large C_{fp}.

Once we obtain some ground truths, such as label consistency or feature distances, a training procedure in Alg. 1 can be used for estimating bit reliability.

```
Data: hash values {h_i} and corresponding labels {l_i}, weights
      $C_{fn}, C_{fp}$
Result: bitReliability
$n = |\{\mathbf{h}_i\}|$, L=length($\mathbf{h}_i$), positiveCounter=$\{0\}^L$;
falsePositiveCounter=$\{0\}^L$, falseNegativeCounter=$\{0\}^L$;
for i=1:n do
    hash₁=$\mathbf{h}_i$, label₁=$l_i$;
    for j=i+1:n do
        hash₂=$\mathbf{h}_j$, label₂=$l_j$;
        truthFlag=(label₁==label₂);
        errorVector=(hash₁ $\oplus$ hash₂);
        positiveFlag=(errorVector==0);
        falsePositiveCounter+=(!truthFlag & positiveFlag);
        falseNegativeCounter+=(truthFlag & !positiveFlag);
        positiveCounter+=truthFlag;
    end
end
negativeCounter=$n \cdot (n-1)/2$-positiveCounter;
falsePositiveRate=falsePositiveCounter/negativeCounter;
falseNegativeRate=falseNegativeCounter/positiveCounter;
bitReliability=$C_{fn}$·falseNegativeRate+$C_{fp}$·falsePositiveRate;
```

Algorithm 1: Estimate bit reliability

C. Tackle bit dependence

So far we have not taken into account the dependence among hash bits. Ideally, a well designed hash algorithm generates independent bits. In practice, bit correlation can be removed to a certain extent by applying orthogonal transforms to the features. However, when a de-correlation stage does not exist, it might be necessary to cope with bit dependence. In this work, we incorporate a graph-based approach. The basic idea is to extract a dense sub-graph (maximal clique). Candidate hash bits are considered as vertices and their dependence is modeled by edges. We define the vertex weight by bit reliability. In addition, the edge weight is defined by the mutual information between a pair of vertex bits. Given the probability $p(b_i)$ for the ith bit and the joint probability $p(b_i, b_j)$ for ith and jth bits, the edge weight measuring the independence between ith and jth bits is empirically computed as:

$$w_{ij} = \exp\{-\alpha \sum_{b_i, b_j} p(b_i, b_j) \log \frac{p(b_i, b_j)}{p(b_i)p(b_j)}\}, \tag{5}$$

where $\alpha > 0$ is a scaling factor. Once we obtain the weighted graph, different optimization approaches can be applied to find a dense sub-graph, such as [18], [13]. In this paper, we adopt the normalized dominant set [15] to discover the optimal solution. In particular, since we define the vertex weight by bit reliability instead of the commonly used similarity preservation (embedding loss), significant performance improvement is achieved, shown and discussed in Section IV-C.

D. Discussion on performance and complexity

It is worth noting that the fusion method, as well as other selection-based methods, is asymptotically guaranteed to be effective, because the probability that the first n bits (i.e. no selection) are the best choice is $1/\binom{n'}{n}$, which is typically very small and decreases with n'.

An advantage of our method is that the reliability of each bit can be individually evaluated. That means the estimation can be realized in a parallel or distributed way. This potentially facilitates bit selection in a very large scale.

Assume the number of training items is N_{tr}. The computation cost for bit reliability estimation increases linearly with the size of the hash bit pool n', and exponentially with N_{tr} (due to pair-wise comparison), so the overall training complexity is approximately $O(\frac{N_{tr}(N_{tr}-1)}{2} \cdot n')$. When N_{tr} is too large, one can reduce the training cost by sampling the training items into groups and performing pair-wise comparison within each group. When there are k groups with N'_{tr} items per group, the total training cost is $O(\frac{N'_{tr}(N'_{tr}-1)}{2} \cdot k \cdot n')$. If we further take into account inter-bit relationship such as mutual information, the additional cost typically increases exponentially with n'.

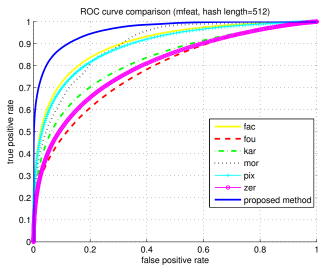

Fig. 2: ROC comparison: feature fusion vs. single features (Case 1, $n = 512$, $n_i = 512$, $n' = 3072$).

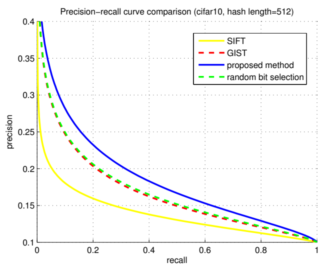

Fig. 3: Precision-recall comparison: feature fusion vs. single features (Case 2, $n = 512$, $n_i = 512$, $n' = 1024$).

The cost for hash generation does not necessarily increase with the size of the hash bit pool, depending on whether we know the internal mechanism of candidate hash algorithms. If we know how each hash bit is generated, then only those good generation functions are kept after bit selection, so that at best the hash generation complexity is always $O(n)$.

IV. Experiment

The experiments include two parts. First, we compare the fusion performance with individual features' performance. Second, we compare our fusion method with other fusion methods. Two application scenarios are considered:

- Case 1 – Handwritten number recognition (mfeat);
- Case 2 – Category based image retrieval (cifar10).

The former is an example of near-duplicate detection; the latter involves more semantics. The results are obtained by plugging LSH into our framework. It is straight-forward to adapt to other hash algorithms.

A. Datasets and experiment setting

Two datasets are used. Case 1 uses the mfeat dataset [2]. It consists of features of handwritten numerals ('0'–'9') extracted from a collection of Dutch utility maps with 200 samples per class (for a total of 2,000 samples). These digits are represented by six features:

1) fac: 216 profile correlations;
2) fou: 76 Fourier coefficients of the character shapes;
3) kar: 64 Karhunen-Loeve coefficients;
4) mor: 6 morphological features;
5) pix: 240 pixel averages in 2 x 3 windows;
6) zer: 47 Zernike moments.

Case 2 uses the cifar10 dataset [12]. It consists of $60,000$ 32×32 color images in 10 classes, with $6,000$ images per class. Each image is represented by two features: 384-D GIST [17] and 300-D bag-of-words using SIFT [16].

Two kinds of tests are considered. For Case 1, receiver operating characteristic (ROC) curves are used as performance benchmarks; for Case 2, precision-recall (P-R) curves are used. Ten rounds of training and testing are performed. In each round, the training set and the testing set are randomly selected. For mfeat, we use $1,000$ images for training and $1,000$ for testing; for cifar10, we use $3,000$ images for training and the whole dataset for testing with $1,000$ randomly selected queries. Average results are presented.

TABLE I: Bit selection from multiple features (Case 1, 10 round average, $n = n_i = 512, n' = 3072$).

Feature name	fac	fou	kar	mor	pix	zer
No. of dimensions	216	76	64	6	240	47
No. of selected bits	133	10	1	312	33	23
Percentage (%)	26.0	2.0	0.2	60.9	6.5	4.5

B. Effects of feature fusion

We first focus on Case 1 and verify the effect of feature fusion using the mfeat dataset. The numbers of feature dimensions sum up to 649. If one would like to utilize all the feature information with a compact representation, a simple approach is to sample some dimensions from each feature. However, since different features have different numbers of dimensions and their significance is unknown with respect to the particular application, it is not straightforward to decide which feature dimensions to select.

We consider a hash length $n = 512$, and apply the proposed framework. The LSH algorithm (Eqn. 2) is used to compute $n_i = 512$ bits from each single feature, resulting in a total of $n' = 3,072$ bits. An n-bit hash value is formed by the proposed method.

Table I lists the results of the bit selection procedure. The last row shows the proportions of features in the fused hash value. There are some interesting observations. First, note that the number of selected bits is not proportional to the corresponding number of feature dimensions. The "fac" and the "pix" features have the most dimensions, but they do not contribute the most bits. The "mor" feature only has 6 dimensions, but provides more than half of the fused hash bits. That means this is a very effective feature. On the other hand, our method only selects 1 bit from the "kar" feature, which is inconsistent with its number of dimensions (64), i.e., there is little information in this feature.

The results of pair-wise hypothesis testing are plotted in Fig. 2. We compare the ROC curve of the proposed method with the curves of individual features. For each feature, the ROC curve is derived from the corresponding 512-bit hash values computed by LSH. We observe that the performance of an individual feature is not proportional to its bit contribution. The "fac" feature has the best overall performance among all the features, followed by "pix". This is somewhat consistent with their top two contribution (26.0% and 6.5%) to the fused hash value. The "mor" feature contributes the most bits, but its performance is only superior for large false positive rates. The "kar" feature contributes the fewest bits (0.2%), but its individual performance is better than "zer" (4.5%) and "fou" (2.0%), which is unexpected. Above all, the proposed method has the best performance. This proves the effectiveness of our fusion framework.

We then look at Case 2. Table II lists the results of bit selection using the cifar10 dataset. We still generate 512 bits from each feature using LSH. The hash bit pool consists of

TABLE II: Bit selection from multiple features (Case 2, 10 round average, $n = n_i = 512, n' = 1024$).

Feature name	sift	gist
No. of dimensions	300	384
No. of selected bits	242	270
Percentage (%)	47.3	52.7

1024 bits. The outcome of bit selection looks more regular than in Case 1. The two features have similar amounts of dimensions (300 vs. 384), and they contribute proportionally (47.3% vs 52.7%). Figure 3 shows a comparison of P-R curves. The "gist" feature performs better than "sift", as predicted by their contribution. Since the two features contribute similarly (242 bits vs. 270 bits), we also show the curve of random bit selection, which randomly selects 256 bits from each feature's hash value. It is interesting that this method almost does not bring any improvement. Again our method is validated. More tests are performed with other hash lengths. Similar trends are obtained, but are not shown due to space limit.

C. Comparison with other fusion methods

Next, we compare our proposal with other fusion methods. Six baselines are considered. Four of them are feature-independent algorithms (early fusion); the other two are bit selection methods (late fusion). Multiple feature kernel hashing (MFKH) [14] is a supervised method that takes multiple features into account. It is set to use 300 anchor points. Spectral hashing (SH) [24] is a well-known unsupervised method based on graph partitioning. Iterative quantization (ITQ) [8] is a recently proposed unsupervised method based on PCA and supervised rotation. The fourth one is LSH [5] itself. For the early fusion methods, the concatenation of feature vectors is used as input. The random selection (RS) method selects approximately an equal amount of bits from each feature's hash value. The normalized dominant set (NDom) method [15] is a recently proposed bit selection based on subgraph extraction, which was the best performing bit selection method to our knowledge.

Figure 4 shows a comparison of ROC curves for Case 1 with hash length $n = 64$. The proposed method exhibits the best performance. Among the rest, NDom takes the lead, followed by RS and ITQ. MFKH and LSH have slightly worse performance than ITQ. It is surprising that SH has the worst performance. This might be due to over-fitting, because mfeat is a small dataset with many feature dimensions. On the other hand, since there are sufficient features, the influence of semantic gaps is diluted, so the untrained LSH shows satisfactory performance. Figure 5 shows another ROC comparison with hash length $n = 128$. The basic trend looks similar, but randomized methods seem to improve. RS now performs as well as NDom, followed by LSH. Above all, the proposed method still performs the best.

Table III shows a performance comparison for other hash lengths. Instead of ROC curves, we estimate the area under the curve, which is an overall measure of performance. Larger values indicate better performance. Our method achieves the largest values for all hash lengths, which proves its superior performance. Among other methods, NDom is generally the best, but it is outperformed by RS at $n = 256$. The good performance of NDom and RS implies that late fusion is better than early fusion in our application. Also note that the performance of SH decreases with n (ITQ exhibits a similar tendency), which means this method does not scale well.

TABLE III: Area under ROC curve at different hash lengths n (Case 1)

n	ITQ	MFKH	SH	LSH	NDom	RS	Ours
32	.8830	.7568	.7860	.6328	.9300	.8577	**.9542**
64	.8542	.8364	.7517	.8362	.9398	.9032	**.9666**
128	.8491	.8298	.6998	.8698	.9384	.9357	**.9691**
256	.8495	.8553	.6561	.8821	.9375	.9461	**.9673**

Figure 6 shows a comparison of P-R curves for Case 2 with hash length $n = 64$. Our method exhibits the best performance, followed by NDom, RS, MFKH, ITQ, LSH, and SH. Since cifar10 is a relatively large dataset with severe semantic gaps, it is not surprising that supervised methods outperform unsupervised early fusion baselines. Unfortunately SH as a trained method still performs poorly. It might be due to the out-of-sample interpolation [24], which is problematic for non-uniformly distributed data. It is interesting that RS works as well as MFKH, which implies the potential advantages of late fusion over early fusion.

Figure 7 shows a comparison of P-R curves for Case 2 with hash length $n = 128$. Similar trends can be observed. Note that for some algorithms the overall performance does not change much with the hash length. This is typically a *semantic* hashing scenario, where the hash value is supposed to carry categorical information which is less than $\lceil \log_2 10 \rceil = 4$ bits. Increasing hash length generally captures more *perceptual* information, which does not necessarily convert to semantics due to the small number (only two) of features. RS now works even better than MFKH, which implies that the effect of supervised bit selection degrades with the hash length.

Table IV shows a performance comparison with other hash lengths. Instead of the P-R curves, we estimate the area under the curve. Compared with Table III, similar trends can be observed. Again our method performs the best.

V. Conclusion

We propose a feature fusion framework for multimedia hashing. Given a feature-independent hash algorithm and several features, hash bits are generated from each feature and put into a pool. A fused hash is generated by selecting best bits out of the pool, which is solved by graph extraction. Specifically, we use bit reliability to represent vertex weights and mutual information to represent edge weights. We instantiate the framework by adopting the classic LSH. Superior performance is confirmed by extensive experiments in two different scenarios, where our method outperforms several state-of-the-art methods.

The proposed framework is generic. Different hash or feature modules can be inserted. The performance may be improved if a better module is used. Our work shows that

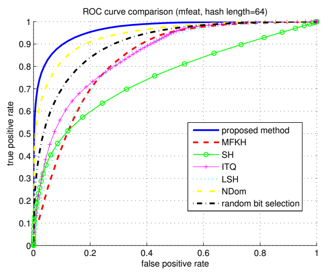

Fig. 4: ROC comparison of fusion methods (Case 1, $n = 64$, $n_i = 512$, $n' = 3072$).

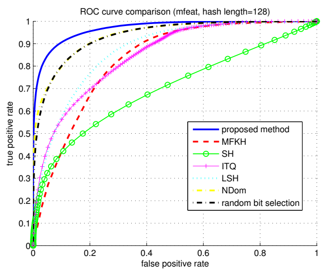

Fig. 5: ROC comparison of fusion methods (Case 1, $n = 128$, $n_i = 512$, $n' = 3072$).

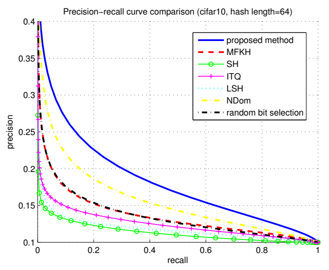

Fig. 6: P-R curve comparison of fusion methods (Case 2, $n = 64$, $n_i = 512$, $n' = 1024$).

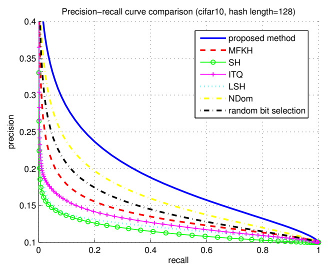

Fig. 7: P-R curve comparison of fusion methods (Case 2, $n = 128$, $n_i = 512$, $n' = 1024$).

TABLE IV: Area under the precision-recall curve at different hash lengths n (Case 2)

n	ITQ	MFKH	SH	LSH	NDom	RS	Ours
32	.1520	.1307	.1293	.1110	.1486	.1274	**.1730**
64	.1252	.1367	.1139	.1188	.1575	.1361	**.1830**
128	.1277	.1391	.1159	.1220	.1642	.1500	**.1896**
256	.1279	.1386	.1160	.1225	.1683	.1587	**.1906**

classic algorithms such as LSH can boost performance and even outperform state-of-the-art by proper utilization.

References

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

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proc. of the 27th International Conference on Machine Learning (ICML)*, pages 111–118, 2010.

[4] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of audio fingerprinting. *Journal of VLSI Signal Processing Systems*, 41(3):271–284, Nov. 2005.

[5] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[6] B. Fernando, E. Fromont, D. Muselet, and M. Sebban. Discriminative feature fusion for image classification. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3434–3441, 2012.

[7] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, Dec 2004.

[8] 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.

[9] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[10] I.-H. Jhuo, S. Gao, L. Zhuang, D. T. Lee, and Y. Ma. Unsupervised feature learning for rgb-d image classification. In *Proc. of Asian Conference on Computer Vision*, pages 276–289, 2014.

[11] I.-H. Jhuo and D. T. Lee. Boosting-based multiple kernel learning for image re-ranking. In *Proc. of ACM International Conference on Multimedia*, pages 1159–1162, 2010.

[12] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.

[13] S. Liu, H. Liu, L. J. Latecki, S. Yan, C. Xu, and H. Lu. Size adaptive selection of most informative features. In *Proc. of AAAI Conference on Artificial Intelligence*, pages 392–297, 2011.

[14] X. Liu, J. He, and B. Lang. Multiple feature kernel hashing for large-scale visual search. *Pattern Recognition*, 47(2):748–757, Feb. 2014.

[15] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1570–1577, 2013.

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

[17] 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.

[18] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):167–172, Jan 2007.

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

[20] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia*, 15(8):1997–2008, Dec 2013.

[21] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[22] S. Voloshynovskiy, O. Koval, F. Beekhof, and T. Holotyak. Binary robust hashing based on probabilistic bit reliablity. In *Proc. of IEEE/SP Workshop on Statistical Signal Processing*, pages 333–336, Aug 2009.

[23] J. Wang, W. Liu, S. Kumar, and S. F. Chang. Learning to hash for indexing big data – a survey. *Proceedings of the IEEE*, 104(1):34–57, Jan 2016.

[24] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. of Conference on Neural Information Processing Systems (NIPS)*, pages 1753–1760, 2008.

[25] 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 (ICME)*, pages 380–385, 2012.

[26] L. Weng, R. Darazi, B. Preneel, B. Macq, and A. Dooms. Robust image content authentication using perceptual hashing and watermarking. In *Proc. of Pacific-Rim Conference on Multimedia*, pages 315–326, 2012.

[27] L. Weng, I.-H. Jhuo, M. Shi, M. Sun, W.-H. Cheng, and L. Amsaleg. Supervised multi-scale locality sensitive hashing. In *Proc. of International Conference on Multimedia Retrieval (ICMR)*, pages 259–266, June 2015.

[28] 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.

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

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

[31] D. H. Wolpert. The relationship between PAC, the statistical physics framework, the bayesian framework, and the VC framework. In *The Mathematics of Generalization*, pages 117–214, 1995.

[32] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):40–51, Jan. 2007.

[33] D. Zhang, X. Liu, and B. Lang. Hash bit selection using markov process for approximate nearest neighbor search. In *Proc. of International Conference on Advances in Mobile Computing & Multimedia (MoMM)*, pages 205–208, 2013.