ON DERIVING LONGER FINGERPRINTS FROM FEATURES BASED ON PROJECTIONS

Regunathan Radhakrishnan, Claus Bauer and Wenyu Jiang

Dolby Laboratories, San Francisco, CA, 94103, USA
Email: regu.r@dolby.com,cb@dolby.com,wzj@dolby.com

ABSTRACT

We propose three methods to derive longer fingerprints from features using projection based hashing methods. For this class of hashing methods, a feature matrix is projected onto a set of projection matrices. Then, the projected values are compared to a threshold to derive the hash bits. The methods proposed in this paper reduce the increase in average number of collisions per hour of content in the database by creating longer fingerprint codewords. The first method derives 56 fingerprint bits by projecting the feature matrix onto 56 projection matrices by deriving one bit from each projected value. The second method uses 36 projection matrices but derives 56 fingerprint bits by deriving varying number of bits from each projected value. The third method also 36 projection matrices but derives 56 bits by creating more features that capture the relationships between the projected values. We present experimental results comparing these three methods to a baseline method in terms of collision performance and robustness.

***Keywords*—** Multimedia fingerprints, Robust Hashing, SVD

1. INTRODUCTION

Robust hash functions are often used for deriving fingerprint bits from features representing the underlying content [1], [2]. The fingerprint bits thus extracted change gradually for changing feature values due to content modifications. Apart from this robustness property of these hash functions, another property that is important for these hash functions is the collision property. A robust hash function is said to have a good collision property if for every $(x \neq y)$, $H(x) \neq H(y)$ with a very high probability. Here x and y are features in a high dimensional space and $H(x)$ and $H(y)$ are fingerprint codewords in low dimensional space. If L is the number of the bits in the fingerprint codeword, then the range space of the hash function has 2^L possibilities. In general, the larger the L more unique (less collisions) are the generated fingerprint codewords.

To further understand the role of L in improving the collision property, let us consider a content identification application with a large database of media fingerprints. Any media fingerprint that is extracted from a query media object is compared against this database of media fingerprints during the identification process. As the size of the database, i.e., the number of fingerprints in the database (N) in terms of number of hours of media increases, it is desirable that the uniqueness of the fingerprint codewords is preserved. The uniqueness property of the fingerprint codewords guarantees that the fingerprint database scales to a large number of fingerprints. Instead, if certain fingerprint codewords are more likely to occur than others, then the uniqueness property vanishes as the database size grows. This leads to an increase in the amount of time spent to find a match. To see this, we consider a hash-table based searching method for matching the query fingerprints against the fingerprints in the database. The database is indexed using the individual fingerprint codewords. Each fingerprint codeword in the hash table points to a location in the database where that fingerprint codeword is present. The number of links per fingerprint index in the hash table will be referred to as number of collisions. If a fingerprint codeword is unique, one can quickly find its match in the database. As the uniqueness reduces, one has to perform more look-ups and pick the best match in terms of smallest hamming distance from the query fingerprint. Thus, fingerprints with a small number collisions per fingerprint codeword will result in a short search duration. Databases containing fingerprints with an average low collision number are more scalable to large database sizes than databases with a high average collision number per fingerprint. Let us consider two content identification applications with two different fingerprint codeword sizes (L_1, L_2 and $L_1 < L_2$). The content identification system with codeword size of L_2 is more likely to have better collision statistics than the one with codeword size of L_1. For instance, if $L_1 = 22$ and the total number codewords (N) in the database is (100*3600*12 = 4320000), then the average number of collisions is 10.29. This is obvious because the range space for hash function with L_1 as codeword size is only $2^{22}(419434)$ which is less than the value of N. On the otherhand, if $L_2 = 34$ and $N = 4320000$, the average number of collisions is 0.00025. In this example, the value of N corresponds to the number of fingerprint codewords in 100hr video database that extracts fingerprint codewords at the rate of 12 frames per second. Industry applications of content identification often require the size of the database to be in the order of hundreds of thousands of hours. For scaling the content identification system to support such large databases, it is always desirable to have a longer fingerprint codeword.

In [1], the authors considered the optimal choice of projection matrices for the robust hash function. Given a training data set for the feature matrix, the proposed method in [1] derives projection matrices that improve collision characteristics. The

projection matrices were obtained using a Singular Value Decomposition (SVD) on the set of features from the training data set. They showed that it is possible to achieve a collision slope (increase in average number of collisions per hour of additional content in the database) of 0.05 using 22 effective fingerprint bits while the collision slope for randomly chosen projection matrices was 1.5. However, the range space of the hash function in [1] with optimized SVD projection matrices is only 2^{22}. It is desirable to have a larger range space for the same hash function to support content identification in large scale media databases.

In this paper, we propose three methods to increase the number of bits per codeword thereby increasing the range space for the hash function. The first method derives more fingerprint bits by projecting the feature matrix onto more projection matrices. It derives one bit from each projected value as in [1]. The second method uses the same set of projection matrices as in [1] but derives more fingerprint bits by quantizing certain robust projections using 4 levels and others using 2 quantization levels. The third method also uses the same set of projection matrices as in [1] but derives additional bits by creating more features that capture the relationships between the projected values. We compare the performance of these three methods in terms of robustness and sensitivity with the performance of the baseline method in [1].

In the following section, we present a brief description of the baseline method in [1].

2. BASELINE METHOD FOR ROBUST HASHING

2.1. Feature Extraction for Video Content

The first step in a video fingerprint extraction process is the derivation of robust features from the video content that are invariant to various processing operations of the content. Let us represent the extracted robust features by a matrix Q. For all the experiments in this paper, Q is a matrix with $G \times N$ elements that attempts to capture the appearance and motion information from the input video. The input video is first temporally downsampled to 12 fps. A letterbox detector detects eventual black bars on the sides of the picture and removes them. Then, every frame is spatially downsampled to $120*160$ ($Height \times Width$) resolution. The N columns in the feature matrix Q correspond to N video frames in a buffer including the current frame. We use a buffer of 3s ($N = 12 \times 3$) to capture motion information. The G rows correspond to a set of moment invariants extracted from each frame in the buffer. We extract 7 moment invariants from 5 concentric circular regions in each frame ($G = 5 \times 7$). Moment invariants are global measures of an image surface that are invariant to translation, rotation and scaling and were originally proposed for text pattern recognition in [3]. Now, each column of the matrix Q attempts to capture the appearance of the corresponding frame of the video in the buffer by measuring how the 7 moment invariants change over the 5 regions. In the second step, hash bits are extracted from the feature matrix Q using a robust hash function. The robust hash function projects the feature matrix (Q) onto L projection matrices P_i , $i = 1, 2, ...L$. Then, each projected value is quantized to obtain the fingerprint bits.

2.2. SVD bases as Projection Matrices for Hashing

In [1], the authors proposed a method to obtain these projection matrices that are optimal in terms of collision performance.

Here, we briefly recall the procedure in the following three steps.

Step 1: Obtain a training dataset of media content and extract features $Q^1, Q^2, ...Q^M$. Here M is the number of training instances. The dimension of each feature matrix Q^i is $R = G \times N$.

Step 2: Then, we compute the covariance matrix C^{feat} (dimension R*R) of the features from the training set $Q^1, Q^2, ...Q^M$ as given by

$$C^{feat}(k,l) = \frac{1}{M}\sum_{i=1}^{M}(Q_k^i - E(Q_k))(Q_l^i - E(Q_l))$$

Here $E(Q_k) = \frac{1}{M}\sum_{i=1}^{M} Q_k^i$ and $k, l = 1, 2...R$.

Step 3: Once we have computed the covariance matrix of the features from the training set, we can now compute the eigenvectors of the matrix C^{feat} that satisfy the relation, $V^{-1}C^{feat}V = D$, using PCA (Principal Components Analysis). Here the columns of V (dimension R*R) are the eigenvectors of the covariance matrix C^{feat} and are represented as $\phi_1, \phi_2, ...\phi_R$. D is a diagonal matrix with eigenvalues $(E_1, E_2...E_R)$ as its main diagonal elements and zero elsewhere.

Now, we transform the input feature vector Q to a L dimensional space by projecting it onto the first L eigenvectors $\phi_1, \phi_2, ...\phi_L$ and obtain $H_1^{svd}, H_2^{svd}, ...H_L^{svd}$ respectively. Here H_k^{svd} is the projection of Q onto ϕ_k. In [1], we used the first 36 significant SVD bases and derived 1 bit per projected value by thresholding (i.e L=36 and fingerprint codeword size is also 36). We showed that the fingerprints derived using this method have lower average number of collisions than fingerprints derived using random projection matrices.

Note that even though the fingerprint size is 36 the effective number of bits per codeword is only 22. This is due to the use of 14 "weakbits" per codeword during matching. "Weakbits" are bit positions that are most likely to flip due to content modifications. During the search procedure, a hash look-up is performed not only for the exact hash codeword but also for the other 2^{14} possible codewords indicated by the weakbits. This reduces the available range space from 2^{36} to 2^{22} though the codeword size is 36.

In the following sections we describe the three proposed methods in this paper to derive more fingerprint bits from the same feature matrix to further increase the range space.

3. PROPOSED METHODS

3.1. Method 1: Using more projection matrices

In [1], the proposed method used the first 36 significant SVD bases as projection matrices. In order to derive 20 more bits from the same feature matrix, we propose to use the first 56 significant SVD bases as projection matrices. Then, the projected values are compared against a threshold to derive a 56-bit fingerprint codeword. We use 22 bits of the 56 bits as weakbits ($(22 = \frac{14}{36} * 56)$the same percentage of weakbits as for a 36-bit fingerprint codeword). Therefore, the effective number of bits per codeword is 34.

Using this method, we cannot derive arbitrarily large number of fingerprint bits by including the corresponding number of significant SVD bases as projection matrices. We need to pay attention to percentage of total energy captured in the first L significant SVD bases that we select as projection matrices. We compute the percentage of energy captured in the first L significant SVD bases using the following equation:

$$Energypercentage = \frac{\sum_{i=1}^{L} E_i}{\sum_{i=1}^{R} E_i}$$

Here E_i is the i^{th} significant eigenvalue and R is the total number of dimensions of the feature matrix. Let us assume that the first L significant eigenvectors capture 90% of the energy, then the remaining $R - L$ eigenvectors capture only 10% of the signal energy. Therefore, if we select certain eigenvectors from the less significant set of $R - L$ eigenvectors as projection matrices, we may be capturing more information about the noise than the feature matrix.

3.2. Method 2: Deriving varying number of bits from projected values

In this method for deriving longer fingerprint codewords, we propose to use the same first 36 significant SVD bases as projection matrices as in [1]. However, instead of deriving just 1 bit per projected value, we derive 2 bits from a set of 20 robust projections and 1 bit from the remaining 16 projections. In order to select the 20 most robust projections, we compute the following Signal-to-Noise-Ratio (SNR) measure from a training set. The training set consists of original videos and their corresponding modified versions. The modifications include compression, cropping, aspect ratio change and rotation. Let us denote the feature matrix extracted from the original video as Q. The feature matrix Q is projected onto the first 36 eigenvectors $\phi_1, \phi_2, ...\phi_{36}$ and obtain $H_1^{svd}, H_2^{svd}, ...H_{36}^{svd}$ respectively. Let us denote the feature matrix extracted from the modified video as $\widetilde{Q}$. The feature matrix $\widetilde{Q}$ is projected onto the the same set of 36 eigenvectors $\phi_1, \phi_2, ...\phi_{36}$ and obtain $\widetilde{H_1^{svd}}, \widetilde{H_2^{svd}}, ...\widetilde{H_{36}^{svd}}$ respectively. Then, the noise on the i^{th} projected value due to content modifications, Z_i, can be computed $Z_i = H_1^{svd} - \widetilde{H_1^{svd}}$. We can compute the Signal-to-Noise-Ratio (SNR) for i^{th} projected value as $\frac{var(H_i^{svd})}{var(Z_i)}$. Finally, we use this measure to sort the projection matrices in descending order and pick the first 20 projections with strongest SNR measure. From the remaining 16 projections, we derive 1 bit.

The quantization thresholds for deriving 2-bits are obtained in a data-driven fashion from the cumulative distribution function (CDF) of the corresponding projected value (H_i^{svd}). Let us denote the CDF of the i^{th} projected value as C_i. Then, we select the following 3 thresholds th_1, th_2, th_3 such that $C_i(th_1) = 0.25, C_i(th_2) = 0.50$ and $C_i(th_3) = 0.75$. Finally, we apply the following quantization rules to obtain 2 bits from the projection H_i^{svd}.

- if $-\infty < H_i^{svd} < th_1$, then generate bits '00'
- if $th_1 < H_i^{svd} < th_2$, then generate bits '01'
- if $th_2 < H_i^{svd} < th_3$, then generate bits '11'
- if $th_3 < H_i^{svd} < \infty$, then generate bits '10'

Note that we use gray code to encode adjacent quantization levels so that the hamming distance between neighboring quantization levels is equal to 1.

3.3. Method 3: Creating meta-features from projected values

In this method for deriving longer fingerprint codewords, we propose to use the same first 36 significant SVD bases as projection matrices and create some more features that capture the relationship between the projected values. The feature matrix Q is projected onto the first 36 eigenvectors $\phi_1, \phi_2, ...\phi_{36}$ to obtain $H_1^{svd}, H_2^{svd}, ...H_{36}^{svd}$ respectively. The first 36 bits of the fingerprint codeword are derived by thresholding each of the projected values. We assign to '1' to i^{th} bit if $H_i^{svd} > 0$ else we assign a '0' to the i^{th} bit.

In order to create 20 more bits, we create the following meta-features V_i as shown below:

$$V_i = \sum_{j=1, i \neq j}^{j=36} (H_i^{svd} - H_j^{svd})^2$$

Here i = 1,2...20. Each V_i captures the relationship of the i^{th} projected value (H_i^{svd}) to the others. Then, we compare each V_i to a threshold to derive the additional 20 bits. The median($V_1, V_2, ...V_{20}$) is used as the threshold.

4. EXPERIMENTAL RESULTS

In this section, we present experimental results on collision and robustness performance of the baseline method and the performance of the other three proposed methods with longer fingerprint codewords. The feature matrix Q for all experiments in this paper is a matrix of dimensions 35×36 ($G \times N$) where 36 corresponds to a 3s buffer of frames at 12fps and 35 corresponds to the 7 moment invariants computed from 5 concentric circular regions of each frame in the buffer. The SVD projection matrices $\phi_1, \phi_2, ..\phi_R$ were obtained using a 94 hr training

Table 1. Collision Performance A: Number of hours in DB Baseline: Increase in avg. num of collisions for Baseline method; M1: Increase in avg. num of collisions for method 1; M2: Increase in avg. num of collisions for method 2;M3 Increase in avg. num of collisions for method 3;

A	Baseline	M1	M2	M3
3.62	0.162	0.000823	0.00188	0.00363
58.35	1.4179	0.018014	0.10598	0.28715
135.00	10.27150	0.047	0.089	0.58743

dataset. Using the baseline method, we extract 36 bit fingerprint codewords (with 14 weakbits) from a 25 min video clip using three sets of projection matrices. Also, using the proposed three methods, we extract 56 bit fingerprint codewords (with 22 weakbits) from the same video clip.

4.1. Collision Performance

In order to study the collision performance of these four methods, first, we add fingerprint codewords extracted from unique content (content that is unrelated to the 25 min clip) to the database. Then, we compute the average number of collisions as we increase the number of fingerprint codewords of unrelated content from 3 hrs upto 135 hrs. Ideally, there should be no change in the average of number of collisions as we add unrelated content to the database. Table 1 summarizes the collision performance of all the 4 methods. The first column (labeled A) shows the number of hours of unrelated content added to the database as we measure the increase in average number of collisions. The second column (labeled Baseline) shows the increase in average number of collisions for the baseline method. Columns labeled M1,M2 and M3 show the same measure for methods 1, 2 and 3 respectively. First, observe that the average number of collisions grows much faster and is larger in magnitude for the baseline method than it does for the other 3 methods. This is as expected and justifies the need for increasing the number of bits in a fingerprint codeword from 36 bits (in the baseline method) to 56 bits (in the proposed methods). Recall that the range space for the baseline method is only 2^{22} and the range space for the other methods is 2^{34} due to the use of weakbits. Figure 1 compares the collision performance of the three methods that derive 56 bit fingerprint codewords. Observe that among the three methods that generate 56 bits, method 1 has a better collision performance than method 2 which in turn has a better collision performance than method 3. In the next subsection, we will compare the robustness performance of these methods.

4.2. Robustness Performance

In order to study the robustness of the fingerprints created by the 4 methods, we created modified versions of the reference content. The attacks on the content include both geometric (rotation,cropping,aspect ratio change) and non-geometric attacks

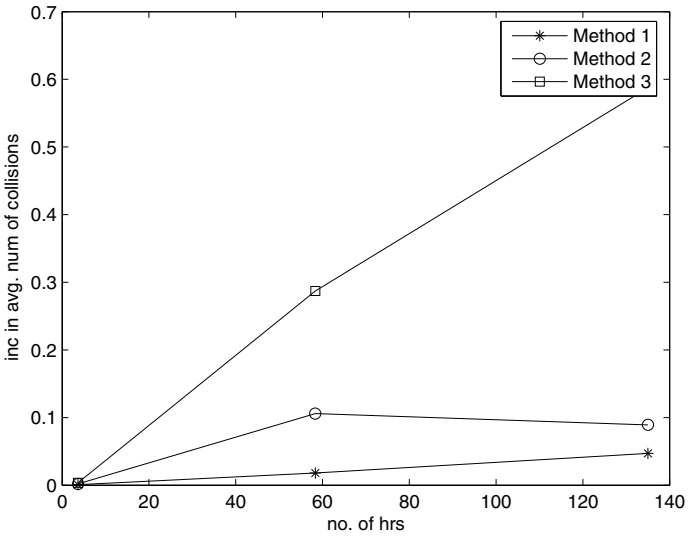

Fig. 1. Comparing collision performance of the methods 1,2 and 3

Table 2. Robustness Performance; Baseline: Avg. BER for Baseline method; M1: Avg. BER for method 1; M2: Avg. BER for method 2; M2: Avg. BER for method 3;

	Baseline	M1	M2	M3
compression	0.0222	0.0295	0.0218	0.0222
rotation	0.0591	0.0716	0.0585	0.0623
cropping	0.1854	0.2084	0.1894	0.1934
bright up	0.0972	0.1089	0.1040	0.1025
bright down	0.1013	0.1089	0.1091	0.1050
asp ratio	0.1000	0.1189	0.1031	0.1088

(compression, brightness change). Then, we compare the fingerprints of the original content to those of the attacked content and record the average Bit Error Rate (BER) for each of the methods. Table 2 summarizes the robustness performance of the 4 methods. Observe row 3 of the table which shows the average BER for cropping attack in the case of all 4 methods. The average BER for cropping is lowest in the case of the baseline method. Method 1 has the highest BER for cropping attack. This is because the Signal-to-Noise ratio is not good for the last 20 SVD projection matrices and they start to capture less information about the signal itself and more information about noise. This is also the reason why Method 1 performs the best in terms of collision statistics. Since noise is generally much more random than actual video content, the derived fingerprint codewords occupy a larger space in the entire range space of the hash function. This results is smaller number of collisions but at the expense of less robustness. Method 2 has similar BER (0.1894) for cropping attacks as the baseline method (0.1854).

4.3. Discussion

Based on the robustness and collision performance results of the 3 proposed methods to create 56-bit fingerprint codewords, we can make the following observations. All three methods have better collision performance than the baseline method with 36-bit fingerprint codewords. this goes to show that in order to support large scale media databases it is useful to derive longer fingerprint codewords. Method 1 which uses 56 projection matrices and derives 1 bit from each projected value, performs the best in terms of collision but performs the worst in terms of robustness. Method 2 has similar robustness performance as the baseline method and has similar collision performance as Method 1. Therefore, Method 2 maintains the robustness of the baseline method while improving the collision performance significantly. Method 2 provides the best trade-off between collision performance and robustness. Method 3 doesn't perform as well as the other two methods either in terms of robustness or in terms of collision. The meta-features seem to be highly correlated and this could be the reason that Method 3's collision performance is a lot worse than those of the other two methods(M1 and M2).

5. CONCLUSION

In this paper, we proposed three methods to derive longer fingerprint codewords from a feature matrix so as to increase the range space of the hash function. We consider the projections based hash method in [1] (which has an effective range space of 2^{22}) as the baseline method and increase its range space to 2^{34}. The first method derives more fingerprint bits by projecting the feature matrix onto more projection matrices. It derives one bit from each projected value as in [1]. The second method uses the same set of projection matrices as in [1] but derives more fingerprint bits by quantizing certain robust projections using 4 levels and others using 2 quantization levels. The third method also uses the same set of projection matrices as in [1] but derives additional bits by creating more features that capture the relationships between the projected values. We compare the performance of these three methods in terms of robustness and sensitivity with the performance of the baseline method in [1]. Based on experimental results, we conclude that method 2 provides the best trade-off between collision performance and robustness. All 3 methods have better collision performance than the baseline method as the effective range space of the proposed methods with longer fingerprint codewords is 2^{34}.

6. REFERENCES

[1] R.Radhakrishnan and C.Bauer, "On improving the collision property of robust hashing based on projections," *Proc. of ICME*, 2009.

[2] J.Fridrich and M.Goljan, "Robust hash functions for digital watermarking," *ITCC*, 2000.

[3] Hu,M.K, "Visual pattern recognition by moment invariants," *IRE Trans. Info. Theory*, vol. IT-8, pp. 179–187, 1962.