LEARNING COMPACT HASHING CODES FOR LARGE-SCALE SIMILARITY SEARCH

BY

HONGHAI YU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

ABSTRACT

Retrieval of similar objects is a key component in many applications. As databases grow larger, learning compact representations for efficient storage and fast search becomes increasingly important. Moreover, these representations should preserve similarity, i.e., similar objects should have similar representations. Hashing algorithms, which encode objects into compact binary codes to preserve similarity, have demonstrated promising results in addressing these challenges. This dissertation studies the problem of learning compact hashing codes for large-scale similarity search. Specifically, we investigate two classes of approach: regularized Adaboost and signal-to-noise ratio (SNR) maximization. The regularized Adaboost builds on the classical boosting framework for hashing, while SNR maximization is a novel hashing framework with theoretical guarantee and great flexibility in designing hashing algorithms for various scenarios.

The regularized Adaboost algorithm is to learn and extract binary hash codes (fingerprints) of time-varying content by filtering and quantizing perceptually significant features. The proposed algorithm extends the recent symmetric pairwise boosting (SPB) algorithm by taking feature sequence correlation into account. An information-theoretic analysis of the SPB algorithm is given, showing that each iteration of SPB maximizes a lower bound on the mutual information between matching fingerprint pairs. Based on the analysis, two practical regularizers are proposed to penalize those filters generating highly correlated filter responses. A learning-theoretic analysis of the regularized Adaboost algorithm is given. The proposed algorithm demonstrates significant performance gains over SPB for both audio and video content identification (ID) systems.

SNR maximization hashing (SRN-MH) uses the SNR metric to select a set of uncorrelated projection directions, and one hash bit is extracted from each projection direction. We first motivate this approach under a Gaussian model for the underlying signals, in which case maximizing SNR is equivalent to minimizing the hashing error probability. This theoretical guarantee differentiates SNR-MH from other hashing algorithms where learning has to be carried out with a continuous relaxation of quantization functions. A globally optimal solution can be obtained by solving a generalized eigenvalue problem. Experiments on both synthetic and real datasets demonstrate the power of SNR-MH to learn compact codes.

We extend SNR-MH to two different scenarios in large-scale similarity search. The first extension aims at applications with a larger bit budget. To learn longer hash codes, we propose a multi-bit per projection algorithm, called SNR multi-bit hashing (SNR-MBH), to learn longer hash codes when the number of high-SNR projections is limited. Extensive experiments demonstrate the superior performance of SNR-MBH. The second extension aims at a multi-feature setting, where more than one feature vector is available for each object. We propose two multi-feature hashing methods, SNR joint hashing (SNR-JH) and SNR selection hashing (SNR-SH). SNR-JH jointly considers all feature correlations and learns uncorrelated hash functions that maximize SNR, while SNR-SH separately learns hash functions on each individual feature and selects the final hash functions based on the SNR associated with each hash function. The proposed methods perform favorably compared to other state-of-the-art multi-feature hashing algorithms on several benchmark datasets.

To my family.

ACKNOWLEDGMENTS

I would like to express my most sincere gratitude and deepest appreciation to my advisor, Prof. Pierre Moulin, for his guidance and support throughout the course of my PhD. It is because of his expertise, understanding, and patience that help me overcome the difficulties during my graduate study. Without his supervision and encouragement, this dissertation would not be possible.

I thank Professors Minh N. Do, Paris Smaragdis, and Venugopal V. Veeravalli for serving on my doctoral committee and their helpful suggestions and advice. A special mention is appropriate for Prof. Kee Chaing Chua and Dr. Sumei Sun for encouraging me to pursue a doctorate. I wish to thank Doctors Sujoy Roy, Stefan Winkler, Bingbing Ni, Jiwen Lu, and Shenghua Gao for illuminating discussion and helpful comments.

It is a pleasure to thank all my research colleagues at the Beckman Institute, Shankar Sadasivam, Yen-Wei Huang, Scott Deeann Chen, Rohit Naini, Ben Chidester, Patrick John-stone, Igor Fedorov, Xinqi Chu, Zhen Li, Jianchao Yang, Shiyu Chang, Sujeeth Bharadwaj, and Huiguang Yang for making my research life memorable with their help and friendship.

My heartfelt thanks go to my parents for their unconditional support, faith, and love. Finally, my special thanks go to my wife, Wen Huang, for the constant encouragement and always staying by my side.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Recent years have witnessed an explosive growth of multimedia data. Every day, more than 350 million photos are uploaded to Facebook.[1] Every minute, 100 hours of video are uploaded to YouTube.[2] On the one hand, these large-scale databases create many opportunities for novel applications. On the other hand, the large volume poses unique challenges for retrieval of similar objects from the database. In particular, any application requiring large-scale similarity search has to address the following challenges:

1. How to define similarity between objects;
2. How to design compact representations so that multimedia objects can be efficiently stored;
3. How to design fast search algorithms so that queries can be evaluated cheaply.

These challenges have motivated the recent studies on hashing methods, where multimedia content is encoded into compact binary hash codes which allows efficient storage and real-time search [1]. The hash codes must be robust to various content-preserving distortions, while being discriminative enough to distinguish perceptually different signals.

Hashing algorithms can be broadly divided into two categories: unsupervised and supervised, depending on how they address the first challenge. In unsupervised hashing, a multimedia signal is represented as a feature vector, and distance (such as Euclidean distance or cosine similarity) between feature vectors is assumed to reflect *semantic similarity*. For instance, images from the same category should have smaller distance between their feature

[1]http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9

[2]Retrieved Dec 27, 2014, from https://www.youtube.com/yt/press/statistics.html

vectors than that of images from different categories. The goal of unsupervised hashing is to design binary codes that preserve a given distance metric in the feature space. Though people continuously search for better features to represent multimedia signals, there still exists a *semantic gap* between feature representations and the richness of human semantics. Often, retrieval performance in the feature space is not satisfactory, yet it provides an upper bound on the performance one would expect with unsupervised hashing algorithms. Locality-sensitive hashing (LSH) [2] and its variants [3, 4], spectral hashing (SH) [5], and iterative quantization (ITQ) [6] are among the most popular and well studied unsupervised algorithms.

Most supervised hashing algorithms also use feature vectors as inputs because it can be difficult to extract good (i.e., robust and discriminative) hash codes directly from high-dimensional raw multimedia signals. However, the goal of supervised hashing is to learn binary codes that preserve semantic labels rather than some distance metric in the feature space. With supervised information, one could potentially obtain better retrieval performance in the Hamming space than that in the feature space, and thus bridge the semantic gap. The superior performance in the Hamming space has been demonstrated in many of our experiments as well as in [7, 8]. Semantic labels have been used in boosting [9], support vector machines (SVMs) [10], and deep neural networks [11] to learn compact binary codes in a supervised manner. Semi-supervised hashing [12] maximizes the empirical accuracy on the labeled data and uses unlabeled data as a regularizer. In Chapter 2, we will study some of the most famous unsupervised and supervised hashing algorithms in more detail.

Note that standard hash functions such as MD5 and SHA-1 cannot be used on multimedia retrieval because the output of such functions changes dramatically with even a single bit change in the input [13]. However, two images or two songs appearing identical to humans can have distinct digital representations. To differentiate from the standard hash functions, hashing for similarity search is often termed robust hashing, semantic hashing, or fingerprinting. Moreover, different from watermarking, which inserts an identifier into the multimedia content and thus changes the content, hashing for multimedia content identification (ID) extracts a signature (fingerprint) from the multimedia content without changing it.

On the information-theoretic side, a framework for fingerprint-based content ID systems was presented in [14], and derived a fundamental relation between database size and query length under some statistical assumptions. Decoding of correlated fingerprints was studied in [15, 16]. The design of scalar quantizers inspired by the notion of biometric identification system capacity [17] was studied in [18]. The related problem of physical object identification was studied in [19].

The goal of this dissertation is to develop better supervised hashing algorithms for large-scale similarity search. We propose two frameworks, namely regularized Adaboost and signal-to-noise ratio (SNR) maximization, to address the aforementioned challenges. Regularized Adaboost, presented in Chapter 3, is motivated by a theoretical study of the boosting framework for content ID, which is one of major applications of large-scale similarity search. The SNR maximization framework, presented in Chapter 4, is motivated by the analysis under a Gaussian model and can be extended into multi-bit hashing in Chapter 5 and multi-feature hashing in Chapter 6. Besides content ID, SNR maximization hashing is applicable to a wider range of applications, such as content-based image retrieval (CBIR). A detailed outline of the rest of the dissertation is provided below.

1.1 Outline of the Dissertation

- **Chapter 2** surveys some of the most popular hashing algorithms, unsupervised and supervised, which we will improve upon or compare with in the rest of the dissertation.

- **Chapter 3** first performs an information-theoretic analysis of the boosting framework and then proposes a regularized Adaboost algorithm aiming at increasing the mutual information between original and degraded fingerprints. A learning-theoretic analysis of the proposed algorithm is also provided, followed by experimental evaluations.

- **Chapter 4** proposes a novel hashing algorithm based on signal-to-noise ratio (SNR) maximization to learn compact binary codes, where the SNR metric is used to select a set of projection directions, and one hash bit is extracted from each projection direction. We first motivate this approach under a Gaussian model for the underlying signals, in which case maximizing SNR is equivalent to minimizing the hashing error probability. A globally optimal solution can be obtained by solving a generalized eigenvalue problem.

- **Chapter 5** develops a multi-bit per projection algorithm to learn longer hash codes when the number of high-SNR projections is limited. We first show the deteriorating effect of low-SNR projections both theoretically and empirically, and propose a remedy which we call SNR multi-bit hashing (SNR-MBH). SNR-MBH is an automatic procedure that determines the number of available high-SNR projections, the number of bits for each projection, and the positions of quantization thresholds. Experiments on a synthetic dataset and real datasets demonstrate the superior performance of SNR-MBH.

- **Chapter 6** proposes two multi-feature hashing methods based on signal-to-noise ratio (SNR) maximization. The first one jointly considers all feature correlations and learns uncorrelated hash functions that maximize SNR, while the second method separately learns hash functions on each individual feature and selects the final hash functions based on the SNR associated with each hash function. The proposed methods perform favorably compared to other state-of-the-art multi-feature hashing algorithms on several benchmark datasets.

- **Chapter 7** reviews and discusses some current research on hashing algorithms, and make two observations within the SNR maximization framework, which we believe could be generalized to other hashing settings and worth further exploration.

  Chapter 8 summarizes the contributions of this dissertation, and discusses future research directions.

CHAPTER 2

BACKGROUND

As introduced in Chapter 1, hashing is a clever way to address the challenges for large-scale similarity search. In hashing, each database item is represented by a compact binary code. The code is constructed such that similar items have similar binary codes. Binary codes are storage efficient and computing Hamming distance can be extremely fast with just a few machine instructions. Millions of database items can be compared to a query in less than a second. Over the last decade, there has been a surge in hashing algorithms, both unsupervised and supervised. In this chapter, we survey some of the most popular hashing algorithms, which we will improve upon or compare with in the rest of the dissertation.

2.1 Unsupervised Hashing Algorithms

When databases are huge, many machine learning tasks such as image scene classification can be performed by a simple nearest neighbor search. However, exhaustively comparing a query with every point in the database may become prohibitively expensive as the database size and feature dimension grow. To reduce search complexity with little performance loss, unsupervised hashing has established itself as an efficient framework for approximate nearest neighbor (ANN) search.

2.1.1 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH), proposed in the late 1990s, was considered a breakthrough for ANN search in high dimensional space and became the paradigm of unsupervised hashing [2]. As depicted in Fig. 2.1a, LSH generates binary codes by randomly projecting data followed by thresholding the projections, and can achieve sublinear search complexity. To preserve locality, each binary hash function $\phi_k : \mathbb{R}^d \to \{\pm 1\}$ must satisfy

$$\Pr\{\phi_k(\mathbf{x}) = \phi_k(\mathbf{y})\} = \mathrm{sim}(\mathbf{x}, \mathbf{y}), \tag{2.1}$$

where $\mathrm{sim}(\mathbf{x}, \mathbf{y}) \in [0, 1]$ is the similarity score in the feature space, e.g., $\mathrm{sim}(\mathbf{x}, \mathbf{y}) = \exp\{-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\}$. A typical class of LSH functions is parameterized by a random projection $\mathbf{w}_k \in \mathbb{R}^d$ and a random threshold $b_k \in \mathbb{R}$:

$$\phi_k(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_k^T \mathbf{x} + b_k), \tag{2.2}$$

where $\mathbf{w}_k$ are sampled randomly from a p-stable distribution, e.g., standard Gaussian, and b_k from a uniform distribution [20].

Although there exists a theoretical guarantee that the Hamming distance between LSH codewords will asymptotically approach the Euclidean distance between the feature vectors [21], it is not very efficient in practice since it requires multiple tables with long codes. Many variants of LSH, such as kernelized LSH [3, 4], have been proposed to exploit the same theoretical guarantee.

2.1.2 Spectral Hashing

In practice, LSH and its variants can lead to very inefficient codes as their hash functions are data independent. To utilize the abundant training data available in large-scale databases, spectral hashing (SH) was proposed to design compact binary codes in a data dependent manner. In addition to the common desired properties of a hash code, i.e., mapping similar samples to similar binary codewords, using a small number of bits to code the full database, and easily computing for a novel point, SH requires the codes to be balanced and uncorrelated [5]. Among all codes that have these properties, SH seeks the ones where the average Hamming distance between similar points is minimal. In particular, SH formulates the hashing problem as the following constrained minimization:

$$\begin{aligned} \underset{\boldsymbol{\phi}}{\text{minimize}} \quad & \sum_{i,j} \text{sim}(\mathbf{x}_i, \mathbf{x}_j) \|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)\|^2 \\ \text{subject to} \quad & \boldsymbol{\phi}(\mathbf{x}_i) \in \{-1, 1\}^J \\ & \sum_i \boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{0} \\ & \sum_i \phi_j(\mathbf{x}_i)\phi_k(\mathbf{x}_i) = 0, \quad \forall j \neq k, \end{aligned} \tag{2.3}$$

where $\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\}$ is defined in (2.1), and $\Phi(\mathbf{x}) = \{\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})\}$ denotes K hash functions.

The authors have noted that solving problem 2.3 with a single bit is equivalent to balanced graph partitioning and is NP hard. The combination of K-bit balanced partition will be even harder because of the pairwise uncorrelated constraints. By relaxing the binary constraint and independent constraint, the above optimization was solved by thresholding eigenvectors of a graph Laplacian. With the assumption of uniform data distribution, the spectral solution can be efficiently generalized to out of samples extension with three key steps [5]: (1) find the maximum variance directions of the data using Principal Component Analysis (PCA); (2) select analytical eigenfunctions using a rectangular approximation along every PCA direction, which prefers dimensions with large range and low spatial frequency; (3) threshold the analytical eigenfunctions at zero.

SH is one of the earliest data dependent hashing algorithms. One major criticism of SH is the assumption of uniform data distribution, which is often too restrictive for practical applications. Following SH, many hashing algorithms have been proposed to better exploit training data for constructing compact binary codes.

2.1.3 Iterative Quantization

Many hashing algorithms encode the PCA directions to generate binary codes [5, 12, 22]. They all note that encoding each direction with the same number of bits leads to poor performance as higher-variance directions carry much more information. To address the imbalanced variance, iterative quantization

(ITQ) [6] aims to balance the variance of different PCA directions by finding a rotation to the PCA-projected data to minimize the quantization error (see Fig. 2.1 for an illustration).

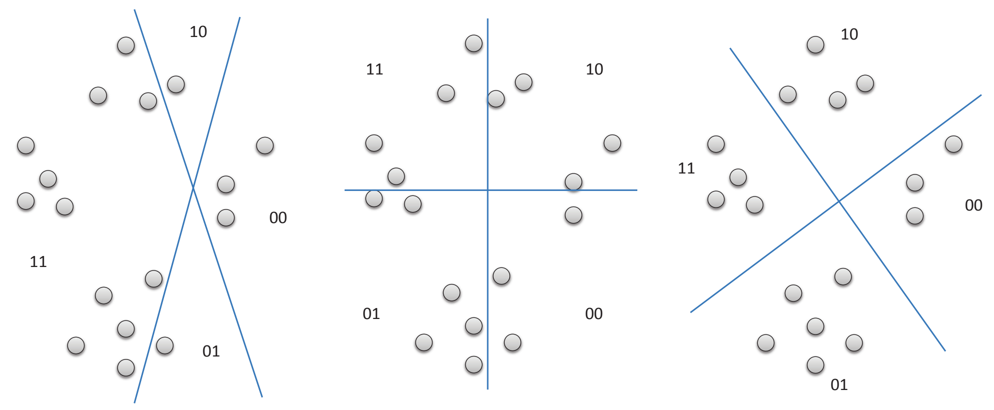
Let $X \in \mathbb{R}^{D\times n}$ denote the zero-centered data matrix with n data points, $W \in \mathbb{R}^{D\times K}$ the PCA matrix with top K eigenvectors of the data covariance matrix XX^T, and $B \in \{-1,1\}^{K\times n}$ the binary code matrix. The goal of ITQ is to learn an orthogonal rotation matrix $R \in \mathbb{R}^{K\times K}$ that minimizes the quantization loss

$$Q(B,R) = \|B - (WR)^T X\|_F^2, \tag{2.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, in two alternating steps.

1. Fix R and update B: $B = \text{sgn}((WR)^T X)$.
2. Fix B and update R: $R = \hat{S}S^T$, where $\hat{S}$ and S are obtained by computing he SVD of the $K \times K$ matrix BX^TW as $S\Omega\hat{S}^T$.

A local optima could be found by alternating between updates to B and R for several iterations. As noted in [6], 50 iterations are often enough to balance the variance and produce good performance.



(a) LSH (b) PCA (c) ITQ

Figure 2.1: Illustration of different hashing.

Recently, product quantization (PQ), a structured vector quantization technique, has been applied to ANN search in high dimensional spaces, where distance between two data points is approximated by the distance between their codewords [23, 24, 25]. For any quantization technique, a large codebook is required to keep the quantization error small and model fidelity high. The strength of a product quantizer is to produce a large effective codebook with the Cartesian product of several small sub-codebooks. ITQ [6] can be considered a special case of PQ. To extract K-bit codes, the codewords are constrained to be taken from the vertexes of a K-dimensional hypercube $\mathcal{C} = \{\pm 1\}^K$, so each sub-quantizer of ITQ is a scalar quantizer (the sign of the projected data). PQ's superior ANN search performance over unsupervised hashing methods is largely due to its adaptive K-means quantization and more precise distance computation than Hamming distance. However, PQ is 10-20 times slower than hashing in search speed [26].

2.2 Supervised Hashing Algorithms

Different from unsupervised hashing, supervised hashing algorithms are not designed to preserve any metric similarity. Instead, semantic labels are used to construct binary codes that have the potential to better preserve the semantic similarity than the original feature vector.

2.2.1 Semi-Supervised Hashing

Semi-supervised hashing (SSH) was proposed to better preserve semantic similarity with labeled data and use unlabeled data as a regularizer [12, 22]. Specifically, SSH was formulated as a maximization of the classification accuracy on the labeled data regularized by data variance over the labeled and unlabeled data. Moreover, SSH considers three different constraints to achieve various degrees of hash bit dependency.

Let $\mathbf{X} \in \mathbb{R}^{D \times n}$ denote the zero-centered data matrix with n data points, in which $l < n$ points are associated with at least one of the categories $\mathcal{T}_+$ or $\mathcal{T}_-$. A pair of points $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{T}_+$ is considered a matching pair, and $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{T}_-$ a nonmatching pair. Let $\boldsymbol{\phi} = \{\phi_1, \ldots, \phi_K\}$ be a sequence of K hash functions parameterized by K projections $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\} \in \mathbb{R}^{D \times K}$, where

$$\phi_k(\mathbf{x}_i) = \mathrm{sgn}(\mathbf{w}_k^T \mathbf{x}_i). \tag{2.5}$$

The goal of SSH is to learn hash functions $\boldsymbol{\phi}$ that maximize the empirical accuracy on the labeled training data:

$$\underset{\boldsymbol{\phi}}{\text{maximize}} \quad \sum_k \left[\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{T}_+} \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{T}_-} \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) \right]. \tag{2.6}$$

To strive for a balanced partition of the projected data, SSH uses the variance of hash bits $\sum_k \text{var}[\phi_k(\mathbf{x})]$ as a regularizer.

Since the sgn$(\cdot)$ function in the objective and regularizer is nondifferentiable, the sign of projection is replaced with signed magnitude. Written in matrix form, the actual objective function of SSH is

$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{2}\text{tr}\left\{\mathbf{W}^T\mathbf{X}_l\mathbf{S}\mathbf{X}_l^T\mathbf{W}\right\} + \frac{\eta}{2}\text{tr}\left\{\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}\right\} \\ &= \frac{1}{2}\text{tr}\left\{\mathbf{W}^T\left[\mathbf{X}_l\mathbf{S}\mathbf{X}_l^T + \eta\mathbf{X}\mathbf{X}^T\right]\mathbf{W}\right\}, \end{aligned} \tag{2.7}$$

where $\mathbf{X}_l \in \mathbb{R}^{D\times l}$ is formed as the l columns of $\mathbf{X}$ containing the l labeled data points, and $\mathbf{S} \in \mathbb{R}^{l\times l}$ is the label matrix which incorporates the pairwise relationship between points from $\mathbf{X}_l$ as:

$$S_{ij} = \begin{cases} 1 & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{T}_+ \\ -1 & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{T}_- \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

With (2.7) as the objective function, the authors considered three different constraints on hash bit dependency, corresponding to three SSH algorithms.

Algorithm 1: Imposing the orthogonality constraint on the projection directions, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, the learning of optimal projections becomes a typical eigen-problem, which can be solved by doing an eigenvalue decomposition on matrix $\mathbf{M} = \mathbf{X}_l\mathbf{S}\mathbf{X}_l^T + \eta\mathbf{X}\mathbf{X}^T$:

$$\mathbf{W}_{\text{orth}} = [\mathbf{e}_1, \ldots, \mathbf{e}_K], \tag{2.9}$$

where $\mathbf{e}_k, k = 1, \ldots, K$ are the eigenvectors of $\mathbf{M}$ corresponding to the top eigenvalues.

Algorithm 2: Similar to the motivation of ITQ, the authors also noted that imposing orthogonality constraint forces one to progressively pick those

directions that have very low variance, substantially reducing the quality of lower bits. Hence, the second algorithm allows subsequent directions to be non-orthogonal with a penalty term:

$$J(\mathbf{W}) = \frac{1}{2}\mathrm{tr}\left\{\mathbf{W}^T\mathbf{M}\mathbf{W}\right\} - \frac{\rho}{2}\|\mathbf{W}^T\mathbf{W} - \mathbf{I}\|_F^2. \tag{2.10}$$

The following procedure was proposed to find a local optimal solution to the nonconvex objective function (2.10).

Choose $\rho > \max(0, -\lambda_{min})$, where λ_{min} is the smallest eigenvalue of $\mathbf{M}$. Then $\mathbf{Q} = \mathbf{I} + \frac{1}{\rho}\mathbf{M}$ will be positive definite. Let $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$ be the Cholesky decomposition. The final non-orthogonal projections are derived as:

$$\mathbf{W}_{nonorth} = \mathbf{L}\mathbf{U}_K, \tag{2.11}$$

where $\mathbf{U}_K$ are the top K eigenvectors of $\mathbf{M}$.

Algorithm 3: The adjustment over the orthogonal ones is done by a single-shot in Algorithm 2. However, the resulting solution is sensitive to the choice of the penalty coefficient. The third algorithm is to learn each projection in a sequential manner. The algorithm is given in Table 2.1, where $\tilde{\mathbf{S}}^k \in \mathbb{R}^{l\times l}$ measures the signed magnitude of pairwise relationships of the k^{th} projections of $\mathbf{X}_l$:

$$\tilde{\mathbf{S}}^k = \mathbf{X}_l^T\mathbf{w}_k\mathbf{w}_k^T\mathbf{X}_l, \tag{2.12}$$

and

$$\mathrm{T}(\tilde{\mathbf{S}}_{ij}^k, \mathbf{S}_{ij}) = \begin{cases} \tilde{\mathbf{S}}_{ij}^k & \mathrm{sgn}(\mathbf{S}_{ij}\cdot\tilde{\mathbf{S}}_{ij}^k) < 0 \\ 0 & \mathrm{sgn}(\mathbf{S}_{ij}\cdot\tilde{\mathbf{S}}_{ij}^k) \geq 0. \end{cases} \tag{2.13}$$

In Algorithm 3, those labeled pairs for which the current hash function predicts their bits wrongly exert more influence on the learning of the next hash function, biasing the new projection to produce correct bits for such pairs. Intuitively, Algorithm 3 has a flavor of boosting-based methods we will introduce next.

Table 2.1: **Algorithm 3** Semi-supervised sequential projection learning for hashing (S3PLH).

```
Input: data $\mathbf{X}$, pairwise labeled data $\mathbf{X}_l$, initial pairwise labels $S_1$, length of hash codes $K$, constant $\alpha$
Do for $k = 1, \ldots, J$
    1. Compute adjusted covariance matrix:
           $\mathbf{M}_k = \mathbf{X}_l \mathbf{S}_k \mathbf{X}_l^T + \eta \mathbf{X}\mathbf{X}^T$
    2. Extract the first eigenvector $\mathbf{e}$ of $\mathbf{M}_k$ and set:
           $\mathbf{w}_k = \mathbf{e}$
    3. Update the labels from vector $\mathbf{w}_k$:
           $\mathbf{S}_{k+1} = \mathbf{S}_k - \alpha \mathrm{T}(\tilde{\mathbf{S}}_{ij}^k, \mathbf{S}_{ij})$
    4. Compute the residual:
           $\mathbf{X} = \mathbf{X} - \mathbf{w}_k \mathbf{w}_k^T \mathbf{X}$
```

2.2.2 Symmetric Pairwise Boosting

Boosting-based hashing algorithms have been applied to pose estimation [9], music identification [27, 28], and video content ID [29]. In this section, we summarize the most recent boosting-based hashing algorithm, *symmetric pairwise boosting* (SPB).

The SPB algorithm [28, 29] operates as follows. A training set $\mathcal{T} \triangleq \{(x_t, y_t, z_t) \in \mathcal{X}^2 \times \{\pm 1\}, t \in \mathcal{T}\}$ is comprised of a subset $\mathcal{T}_+$ of $|\mathcal{T}|/2$ *matching pairs* and a subset $\mathcal{T}_-$ of $|\mathcal{T}|/2$ *nonmatching pairs*, where a pair $(x_t, y_t) \in \mathcal{X}^2$ is said to be matching if the second signal is a distorted version of the first, and nonmatching if the two signals are independent. The binary variable (label) z_t is equal to 1 (resp. -1) if (x_t, y_t) is matching (resp. nonmatching). Define a set of J *weak classifiers* $h_j : \mathcal{X}^2 \to \{\pm 1\}, 1 \le j \le J$, as

$$h_j(x, y) = \begin{cases} +1 & \text{if } \phi_j(x) = \phi_j(y) \\ -1 & \text{otherwise,} \end{cases} \tag{2.14}$$

where ϕ_j is parameterized by a filter $\lambda_j : \mathcal{X} \to \mathbb{R}$ and a quantizer $Q_j : \mathbb{R} \to \mathcal{A}$,

$$\phi_j(x) = Q_j(\lambda_j(x)). \tag{2.15}$$

Denote by $\mathcal{H}$ the class of feasible classifiers (indexed by the choice of filters and quantizers).

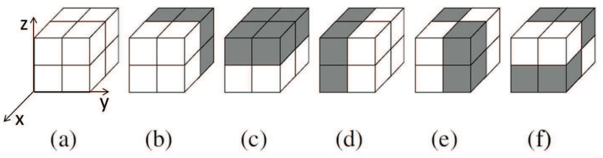


Figure 2.2: 3-D Haar-like filters [29]: (a) spatio-temporal average, (b) temporal difference, (c,d) spatial difference, and (e,f) spatio-temporal difference. The x-coordinate is video frame index.

A popular family of filters is the Haar-like Viola-Jones filters used in [27, 28, 29] which are easy to compute and rich enough to describe perceptually significant visual features. The filter outputs for the 3-D Haar-like filters in [29] are the average difference between values in light and dark regions shown in Fig. 2.2.

To reduce the computational complexity of the training, a limited number of candidate quantizers are evaluated. In [28], 19 candidate thresholds that minimize the mean squared quantization error of the filter responses of the training data are considered. In [29], 17 logarithmically spaced candidate thresholds are considered. For 4-level quantization, $|\mathcal{A}| = 4$, 969 and 680 candidate quantizers are evaluated for each filter for 19 and 17 candidate thresholds respectively.

The SPB algorithm is an adaptation of the well-known Adaboost classification algorithm given in Table 2.2. Upon completion of the algorithm, Adaboost would output the *boosted classifier*

$$h_{\mathrm{B}}(x,y) \triangleq \operatorname{sgn}\left[\sum_{1\le j\le J} \alpha_j h_j(x,y)\right].$$

However the algorithms of [28, 29] do not use the boosted classifier. Only the filter λ_j and quantizer Q_j associated with each h_j are used to produce the fingerprints. The weights $\{\alpha_j\}$ could be used to compute a weighted Hamming distance $D(\underline{f}, \underline{g}) = \sum_{j=1}^{J} \alpha_j d_H(f_j, g_j)$, where d_H denotes the Hamming distance. However, in our content ID experiments, decoders based on Hamming and weighted Hamming distances yield similar results, whereas computing weighted Hamming distance is considerably slower. Thus, we simply report results for Hamming distance.

Table 2.2: Adaboost for filter and quantizer selection.

```
Input: training set $\mathcal{T} \triangleq \{(x_t, y_t, z_t) \in \mathcal{X}^2 \times \{\pm 1\}, t \in \mathcal{T}\}$
Initialization: define equal weights $w_t^{(1)} = 1/|\mathcal{T}|, \forall t \in \mathcal{T}$
Do for $j = 1, \ldots, J$
    1. Choose the classifier $h_j$ that minimizes the weighted error over $h \in \mathcal{H}$
           $e_j = \sum_{t \in \mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) \neq z_t\}.$    (2.16)
    2. Compute $\alpha_j = \frac{1}{2} \log \frac{1-e_j}{e_j}$.
    3. Update the weights
           $w_t^{(j+1)} = w_t^{(j)} \exp\{-\alpha_j z_t h_j(x_t, y_t)\}.$
    4. Normalize the weights so that $\sum_{t \in \mathcal{T}} w_t^{(j+1)} = 1$.
Output: $J$ pairs of filter and quantizer $\{(\lambda_j, Q_j)\}_{j=1}^J$ parameterizing the chosen $J$ classifiers $\{h_j\}_{j=1}^J$.
```

CHAPTER 3

REGULARIZED ADABOOST FOR CONTENT IDENTIFICATION

Content identification (ID) has received considerable attention from both academia and industry. For instance, YouTube uses content ID to detect registered audio and video uploads in real time. Shazam and SoundHound use content ID for music identification on mobile devices. Other applications include advertisement tracking, broadcast monitoring, copyright control, and law enforcement [30, 31, 32, 33]. In these applications, the content is encoded into a short fingerprint which allows for real-time search. The fingerprint must be robust to various content-preserving distortions, while being discriminative enough to distinguish perceptually different signals. The fingerprint is also known as a robust hash, or semantic hash.

As illustrated in Fig. 3.1, a typical content ID system takes a snippet of a signal as a query and seeks a match in a fingerprint database. The system can be broken down into an offline part and an online part. The fingerprint database is built offline by extracting fingerprints from all database signals. When a query comes in, its fingerprint is extracted and used as a query in the fingerprint database.

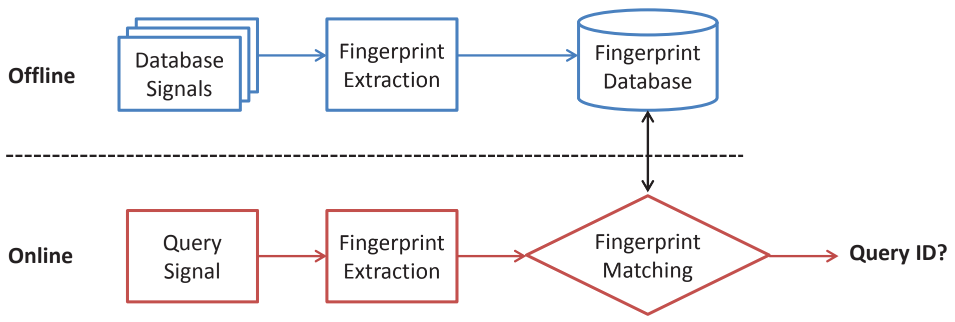


Figure 3.1: Overview of a video content ID system.

3.1 Statement of the Content ID Problem

Following [14], a *content database* is defined as a collection of M elements, $\mathbf{x}(m) \in \mathcal{X}^N, m = 1, 2, \ldots, M$, each of which is a sequence of N *slices* $\{x_1(m), x_2(m), \ldots, x_N(m)\}$. A slice could be a short video clip, a short sequence of image blocks, or a short audio snippet. Slices may be overlapping spatially, temporally, or both, to prevent misalignment during identification [30]. For instance, the video fingerprinting paper [29] uses overlapping time windows that are 1 sec long and start every 100 ms; the temporal overlap is 9/10. A 3-minute video is represented by $N = 1791$ slices. It is desired that the video be identifiable from a short clip, say 10 sec long, corresponding to $L = 91$ slices. This is called the *granularity* of the video ID system [29]. Typically $L \ll N$.

The problem is to determine whether a given *query* consisting of $L < N$ slices, $\mathbf{y} \in \mathcal{X}^L$, is related to some element of the database, and if so, identify which one. To this end, an algorithm $\psi(\cdot)$ must be designed, returning the decision $\psi(\mathbf{y}) \in \{0, 1, 2, \ldots, M\}$, where $\psi(\mathbf{y}) = 0$ indicates that $\mathbf{y}$ is unrelated to any of the database elements. This is a *single-output decoder.* Alternatively, a *variable-size list decoder* $\mathcal{L}(\mathbf{y}) \subseteq \{1, 2, \ldots, M\}$ might be used, returning 0, 1, 2 or more matches.

For applications where a unique output is desirable, such as the YouTube content ID system, the single-output decoder is used. The variable-size list decoder is useful for applications, such as the SoundHound music identification, that can tolerate a few incorrect items as long as the correct one is on the list. To comprehensively analyze the proposed algorithm, we will consider both decoders for our experiments in this chapter.

3.2 Performance Metrics

Different performance metrics are considered depending on which decoder is used. Two types of error, namely false positive and false negative, are associated with the single-output decoder $\psi(\mathbf{y})$. In [14], the content ID problem is viewed as a hypothesis testing problem with $M + 1$ hypotheses $H_0, H_1, \ldots, H_M$, where the null hypothesis H_0 indicates the query is unrelated to any of the database item, the probability of false positive is

$$P_{FP} \triangleq Pr[\psi(\mathbf{Y}) > 0 | H_0], \tag{3.1}$$

and the probability of false negative is

$$P_{FN} \triangleq \frac{1}{M} \sum_{m=1}^{M} Pr[\psi(\mathbf{Y}) \neq m | H_m]. \tag{3.2}$$

There are two error events of interest for the variable-size list decoder of $\mathcal{L}(\mathbf{Y})$:

- *Miss*: The correct m does not appear on the decoder's list, $m \notin \mathcal{L}(\mathbf{Y})$.
- *Incorrect Decoding*: One or more incorrect $m' \neq m$ appear on the decoder's list, $m' \in \mathcal{L}(\mathbf{Y})$. The number of incorrect items on the list is

$$N_{\mathrm{i}}(m) \triangleq \sum_{\substack{m' \neq m \\ 1 \leq m' \leq M}} \mathbb{1}\{m' \in \mathcal{L}(\mathbf{Y}) | H_m\}.$$

Corresponding to these two events are the probability of miss:

$$P_{miss} \triangleq \frac{1}{M} \sum_{m=1}^{M} Pr[m \notin \mathcal{L}(\mathbf{Y}) | H_m] \tag{3.3}$$

and the expected number of incorrect items on the list:

$$\begin{aligned} \mathbb{E}[N_{\mathrm{i}}] &\triangleq \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}[N_{\mathrm{i}}(m)] \qquad (3.4) \\ &= \frac{1}{M} \sum_{m=1}^{M} \sum_{\substack{m' \neq m \\ 1 \leq m' \leq M}} Pr[m' \in \mathcal{L}(\mathbf{Y}) | H_m]. \end{aligned}$$

3.3 Structured Content ID Codes

In this chapter, we restrict our attention to the following fairly general class of fingerprint-based content ID codes. The codes of [30, 27, 28, 29] among others, fall in this category.

Definition 1. *A (M, N, L) structured content ID encoder for a size-M database populated with $\mathcal{X}^N$-valued content items, and granularity L, is a mapping $\phi : \mathcal{X} \to \mathcal{F}$ generating an encoding function $\Phi : \mathcal{X}^N \to \mathcal{F}^N$ that returns a fingerprint $\boldsymbol{f} = \Phi(\boldsymbol{x})$ with components $\underline{f_i} = \phi(x_i)$ for each $1 \le i \le N$.*

Hence the mapping ϕ is applied independently to each slice. It might be convenient to impose additional structure on the code. For instance, the mapping $\phi : \mathcal{X} \to \mathcal{F}$ in [28, 29] is obtained by applying a set of J optimized filters to each slice and quantizing each of the J real-valued filter outputs to four levels. Hence $\mathcal{F}$ takes the form $\mathcal{A}^J$ with $\mathcal{A} = \{a, b, c, d\}$. In this case we view the fingerprint as an array $\mathbf{f} = \{f_{ij}, 1 \le i \le N, 1 \le j \le J\}$ and the query fingerprint as an array $\mathbf{g} = \{g_{ij}, 1 \le i \le L, 1 \le j \le J\}$ where i denotes time and j filter index. We also use the notation $\underline{f} = \{f_j, 1 \le j \le J\}$ for the subfingerprint associated with a given slice. We also write ϕ in vector form as $\phi = \{\phi_j, 1 \le j \le J\}$. The length of the binary subfingerprint $\underline{f}$ is $J \log_2 |\mathcal{A}|$.

In most content ID systems, the decoding function is constructed from a distance measure between fingerprints [28, 29, 14, 16]. Define a decoding metric $d : \mathcal{F}^2 \to \mathbb{R}$, extended additively to pairs of subfingerprints $\{\underline{f_{i+N_0}}, \underline{g_i}, 1 \le i \le L\}$ at time offset $N_0 \in \{0, 1, 2, \ldots, N-L\}$ as follows:

$$d(\mathbf{f}, \mathbf{g}|N_0) \triangleq \sum_{i=1}^{L} d(\underline{f_{i+N_0}}, \underline{g_i}). \tag{3.5}$$

In this chapter, the Hamming metric is used to allow a fair comparison with the SPB algorithm of [28, 29]. Further define the distance between the query fingerprint $\mathbf{g}$ and the database fingerprint $\mathbf{f}$ as the minimum over N_0

$$d^*(\mathbf{f}, \mathbf{g}) \triangleq \min_{N_0} d(\mathbf{f}, \mathbf{g}|N_0). \tag{3.6}$$

Based on the decoding metric d, the two decoders are defined as follows for a decision threshold τ.

Definition 2. *The single-output decoder is*

$$\psi(\boldsymbol{g}) \triangleq \begin{cases} m & \textit{if } d^*(\boldsymbol{f}(m), \boldsymbol{g}) < \tau \textit{ and } d^*(\boldsymbol{f}(m), \boldsymbol{g}) < d^*(\boldsymbol{f}(m'), \boldsymbol{g}), \forall m' \neq m \\ 0 & \textit{if no such } m \textit{ exists.} \end{cases} \tag{3.7}$$

When the minimizer of $d^*(\mathbf{f}(m), \mathbf{g})$ is not unique, the single-output decoder declares no match ($\psi(\mathbf{g}) = 0$). Ties could also be broken at random, but returning a single incorrect match is often more costly than returning no match.

Definition 3. *The variable-size list decoder is*

$$\mathcal{L}(\boldsymbol{g}) \triangleq \{m \in \{1, 2, \ldots, M\} : d^*(\boldsymbol{f}(m), \boldsymbol{g}) < \tau\}. \tag{3.8}$$

Note that $\mathbb{E}[N_\mathrm{i}]$ can be greater than one, whereas $P_{FP} \leq 1$. Moreover, we have $P_{miss} \leq P_{FN}$ as shown in Appendix A. The decision threshold τ is associated with a point on the receiver operating characteristic (ROC) curve, and is chosen according to the desired false positive / false negative error probability tradeoff.

3.4 Mutual Information between Fingerprints

In this section we first review the relevance of mutual information for fingerprint code design, then establish a connection to Adaboost, and finally set up a framework for handling temporal dependencies within fingerprint sequences.

3.4.1 Mutual Information and Content ID capacity

A content ID system, like any other communication system, is subject to a fundamental capacity limit that upper bounds the rate at which information can be decoded with arbitrarily low probability of error. The content ID capacity is the supremum of all fingerprint code rates such that both $\mathbb{E}[N_\mathrm{i}]$ and P_{miss} vanish as $L \to \infty$ [14]. For an i.i.d. signal process $\mathbf{X}$, memoryless degradation channel, and fixed structured content ID code (Def. 1 and Def. 3) with mapping $\phi : \mathcal{X} \to \mathcal{F}$, the content ID capacity is given by $C = I(\underline{F}; \underline{G})$

[14], where $\underline{G}$ is a distorted version of fingerprint $\underline{F}$ and is stochastically related to $\underline{F}$ via the conditional probability distribution $P_{\underline{G}|\underline{F}}$. If ϕ is a code design parameter, then $C = \max_\phi I(\underline{F}; \underline{G})$. Roughly speaking, the largest database that can be handled is $M \approx 2^{LC}$. When the signal $\mathbf{X}$ is an ergodic stationary process and the degradation channel from $\mathbf{X}$ to $\mathbf{Y}$ is stationary ergodic, we propose to use the closely related design criterion $C_L = \max_\phi \frac{1}{L} I(\mathbf{F}; \mathbf{G})$.

The normalized mutual information, $\frac{1}{L} I(\mathbf{F}; \mathbf{G})$, is a nondecreasing function of the number of classifiers J, which is fixed here. Furthermore, in the analysis of Adaboost-based fingerprinting methods, we make the reasonable assumption that the mutual information is approximately additive over filters, i.e., $I(\mathbf{F}; \mathbf{G}) \approx \sum_{j=1}^{J} I(\mathbf{F}_j; \mathbf{G}_j)$. This assumption is justified by the near-independence between learned filters for both SPB and regularized Adaboost.

3.4.2 Information-Theoretic Analysis of SPB

We now show that at each iteration $1 \le j \le J$, SPB maximizes a lower bound on the mutual information $I(F_j; G_j) = H(F_j) - H(F_j|G_j)$ associated with the joint probability distribution $P_{F_j G_j}$ induced by the choice (2.15) of ϕ_j. Indeed we may rewrite (2.16) as follows. SPB selects the weak classifier that minimizes the weighted error

$$h_j = \arg\min_{h \in \mathcal{H}} \left[\sum_{t \in \mathcal{T}_+} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) = -1\} + \sum_{t \in \mathcal{T}_-} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) = 1\} \right], \quad (3.9)$$

where the two error terms are the empirical weighted *false-negative* and *false-positive* error probabilities, respectively. For a given classifier $h \in \mathcal{H}$, the empirical version of the false-negative error probability for matching fingerprints, $P_{e,j} = P_{F_j G_j}(F_j \neq G_j)$, is given by

$$\widehat{P_{e,j}} = \widehat{\Pr}(F_j \neq G_j | \mathcal{T}_+, h) = \sum_{t \in \mathcal{T}_+} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) = -1\}, \quad (3.10)$$

and the empirical false-positive error probability, $P_{F_j} P_{G_j}(F_j = G_j)$, is

$$\widehat{\Pr}(F_j = G_j | \mathcal{T}_-, h) = \sum_{t \in \mathcal{T}_-} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) = 1\}. \quad (3.11)$$

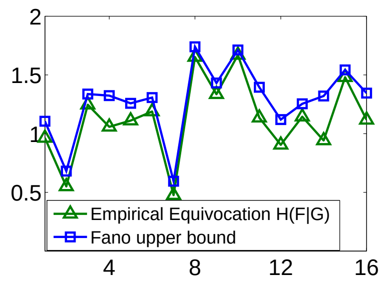
First, we derive a link between $\widehat{\Pr}(F_j \neq G_j | \mathcal{T}_+, h)$ and $\widehat{H}(F_j|G_j)$. By Fano's inequality [34]

$$H(F_j|G_j) \leq h_2(P_{e,j}) + P_{e,j} \log(|\mathcal{A}| - 1), \tag{3.12}$$

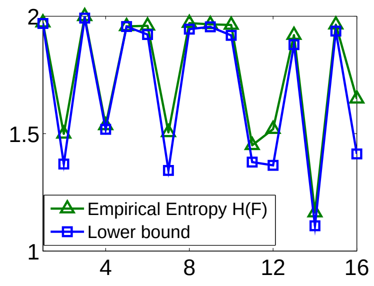
where $P_{e,j} \triangleq P_{F_jG_j}(F_j \neq G_j)$, $\mathcal{A}$ is the alphabet for F_j, and $h_2(P_{e,j})$ is the binary entropy function. One may expect that a similar inequality holds using the empirical version of $H(F_j|G_j)$ and $P_{e,j}$:

$$\widehat{H}(F_j|G_j) \lesssim h_2(\widehat{P_{e,j}}) + \widehat{P_{e,j}} \log(|\mathcal{A}| - 1). \tag{3.13}$$

We have observed empirically that inequality (3.13) is tight. Fig. 3.2a shows the empirical equivocation $\widehat{H}(F_j|G_j)$ and Fano's upper bound $h_2(\widehat{P_{e,j}}) + \widehat{P_{e,j}} \log(|\mathcal{A}| - 1)$ evaluated from 16,000 matching pairs and 16 classifiers. In view of (3.10) and (3.13), minimizing $\widehat{\Pr}(F_j \neq G_j | \mathcal{T}_+, h)$ is tantamount to minimizing a tight upper bound on the empirical conditional entropy $\widehat{H}(F_j|G_j)$.



(a) $\widehat{H}(F_j|G_j)$ and $h_2(\widehat{P_{e,j}}) + \widehat{P_{e,j}} \log(|\mathcal{A}| - 1)$.



(b) $\widehat{H}(F_j)$ and $-\log \widehat{\Pr}(F_j = G_j)$.

Figure 3.2: Simulation results of (3.13) and (3.15). The x-coordinate is the classifier index j.

Next, we derive a link between $\widehat{\Pr}(F_j = G_j | \mathcal{T}_-, h)$ of (3.11) and $\widehat{H}(F_j)$. When F_j and G_j are generated from nonmatching pairs, we model them by a product distribution with identical marginals. From Lemma 2.10.1 in [34], we have $P_{F_j}P_{G_j}(F_j = G_j) \geq 2^{-H(F_j)}$, for two i.i.d. random variables F_j and G_j. Hence $H(F_j)$ is lower bounded by

$$H(F_j) \geq -\log P_{F_j}P_{G_j}(F_j = G_j). \tag{3.14}$$

The empirical version of (3.14) would be

$$\widehat{H}(F_j) \gtrsim -\log \widehat{\Pr}(F_j = G_j | \mathcal{T}_-, h). \tag{3.15}$$

Again, we have observed empirically that (3.15) holds and is tight from non-matching pairs, as shown in Fig. 3.2b. Thus, minimizing $\widehat{\Pr}(F_j = G_j | \mathcal{T}_-, h)$ is tantamount to maximizing a tight lower bound on $\widehat{H}(F_j)$.

From the above argument, we conclude that each iteration $1 \leq j \leq J$ of SPB simultaneously minimizes an upper bound on $H(F_j|G_j)$ and maximizes a lower bound on $H(F_j)$, thus maximizing a lower bound on $I(F_j; G_j) = H(F_j) - H(F_j|G_j)$.

3.4.3 Temporal Dependencies

In content ID systems, slices are temporally overlapped to overcome misalignment during identification, which results in temporally correlated fingerprints $\mathbf{F}_j = \{F_{1j}, F_{2j}, \ldots, F_{Lj}\}$ for each chosen classifier h_j. In a memoryless channel where each G_{ij} only depends on $\mathbf{F}_j$ only via F_{ij}, we have [34]

$$I(\mathbf{F}_j; \mathbf{G}_j) \leq \sum_{i=1}^{L} I(F_{ij}; G_{ij}). \tag{3.16}$$

Equality holds when the input components $\{F_{1j}, F_{2j}, \ldots, F_{Lj}\}$ are independent. Conversely,

$$I(\mathbf{F}_j; \mathbf{G}_j) \ll \sum_{i=1}^{L} I(F_{ij}; G_{ij}), \tag{3.17}$$

when $\{F_{1j}, F_{2j}, \ldots F_{Lj}\}$ are highly correlated. Thus we can increase the mutual information by decorrelating temporal fingerprints. Many slice-wise distortions can be modeled as memoryless channels, including resizing, cropping and rotation.

In the next section, we show that the classifiers' ability to decorrelate slices differs dramatically across different types of filters. In order to increase mutual information by decorrelating temporal fingerprints, we propose to use a regularizer to effectively eliminate from the candidate pool $\mathcal{H}$ those filters that generate highly correlated fingerprints. Experiments demonstrate the effectiveness of this regularizer.

3.5 Regularized Adaboost

A shortcoming of Adaboost for filter selection is the implicit assumption that slices are drawn independently from some unknown distribution. In practice, slice overlapping is necessary to overcome misalignment during identification. For instance, the papers [28] and [29] use overlapping factors of 10/11 and 9/10 respectively. Then slices are significantly correlated. In this section, we propose two regularizers to improve the content ID performance of the Adaboost algorithm in Table 2.2.

3.5.1 Mutual Information of Gauss-Markov Process as a Regularizer

The first regularizer we propose is based on a first-order stationary Gauss-Markov process model for the filter response $\lambda(\mathbf{X})$. The statistical structure of the centered process is completely determined by the correlation coefficient between two consecutive samples $\lambda(X_i)$ and $\lambda(X_{i+1})$. Equivalently, the process is characterized by the mutual information between $\lambda(X_i)$ and $\lambda(X_{i+1})$ [34]

$$I(\lambda) = -\frac{1}{2}\log(1-\rho^2), \tag{3.18}$$

where $\rho \in [-1, 1]$ is the correlation coefficient between $\lambda(X_i)$ and $\lambda(X_{i+1})$. The functional $I(\lambda)$ captures the filter's ability to decorrelate consecutive slices and can be easily estimated from the training dataset. In Fig. 3.3, we show the estimated $I(\lambda)$ for the family of Haar-like filters of Fig. 2.2 applied to video data. Within the family, type (b), (e) and (f) filters compute temporal differences and decorrelate temporal overlapping slices extremely well. Type (a) filters compute the average pixel intensity across the 3-D volume and produce highly correlated responses due to high temporal overlapping. For the spatial difference filters, type (d) filters produce less correlated responses than type (c) filters because horizontal camera movement is normally more frequent than vertical movement, resulting in more spatial difference in the horizontal direction.

From (3.18), we see that for small $|\rho|$, $I(\lambda)$ can be approximated by a linear function of $|\rho|$, while for large $|\rho|$, $I(\lambda)$ increases much faster than linearly. We penalize filters with large mutual information between consecutive output

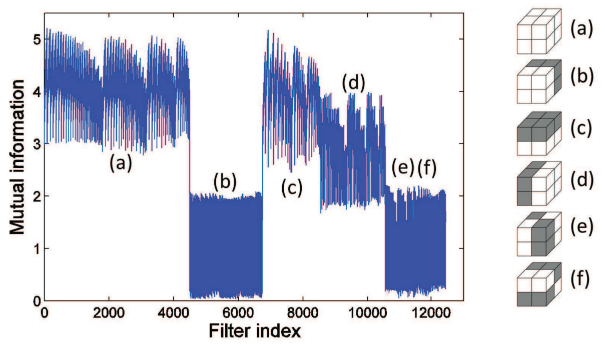


Figure 3.3: Mutual information $I(\lambda)$ for the family of Haar-like filters on video slices.

samples using the new objective function

$$e_j^{\mathrm{REG}} = \sum_{t \in \mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) \neq z_t\} + \gamma I(h), \tag{3.19}$$

where $I(h) = I(\lambda)$ indicates the weak classifier h is parameterized by the filter λ, and $\gamma \geq 0$ is the regularization parameter which can be chosen by cross validation. If $\gamma = 0$, filters are selected by their weighted error on the training dataset without considering their ability to decorrelate overlapping slices. If $\gamma \to \infty$, filters are chosen solely based on $I(h)$. By varying the parameter γ, one can explore the tradeoff between a filter's ability to classify a single slice and to decorrelate overlapping slices. The mutual information regularized (MIR) Adaboost is given in Table 3.1 by replacing $R(h)$ with $I(h)$.

3.5.2 Average Correlation Coefficient as a Regularizer

Our second regularizer makes no Markovian or stationarity assumption about the filter response process. For a given filter λ (such as those from Fig. 2.2), the response $\lambda(\mathbf{X}) = \{\lambda(X_i), 1 \leq i \leq L\}$ is an L-dimensional random vector. Denote by $\rho(s,t) \in [-1,+1]$ the correlation coefficient between two random variables $\lambda(X_s)$ and $\lambda(X_t)$. Define the average correlation coefficient (ACC) of $\lambda(\mathbf{X})$ as

$$\overline{\rho}(\lambda) \triangleq \frac{1}{L^2 - L} \sum_{s \neq t} |\rho(s,t)|. \tag{3.21}$$

The functional $\overline{\rho}(\lambda)$ captures the filter's ability to decorrelate overlapping

Table 3.1: Regularized Adaboost for filter and quantizer selection. $R(h)$ is a generic regularizer. We use $R(h) = I(h)$ for MIR Adaboost and $R(h) = \overline{\rho}(h)$ for ACCR Adaboost.

```
**Input**: training set $\mathcal{T} \triangleq \{(x_t, y_t, z_t) \in \mathcal{X}^2 \times \{\pm 1\}, t \in \mathcal{T}\}$
**Initialization**: define equal weights $w_t^{(1)} = 1/|\mathcal{T}|, \forall t \in \mathcal{T}$
**Do for** $j = 1, \dots, J$
    1. Choose the classifier $h_j$ that minimizes the weighted error over $h \in \mathcal{H}$
       $e_j^{\text{REG}} = \sum_{t \in \mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) \neq z_t\} + \gamma R(h).$   (3.20)
    2. Compute $\alpha_j = \frac{1}{2} \log \frac{1 - e_j^{\text{REG}}}{e_j^{\text{REG}}}$.
    3. Update the weights
       $w_t^{(j+1)} = w_t^{(j)} \exp\{-\alpha_j z_t h_j(x_t, y_t)\}$.
    4. Normalize the weights so that $\sum_{t \in \mathcal{T}} w_t^{(j+1)} = 1$.
**Output**: $J$ pairs of filter and quantizer $\{(\lambda_j, Q_j)\}_{j=1}^{J}$ parameterizing the chosen $J$ classifiers $\{h_j\}_{j=1}^{J}$.
```

slices and can be easily estimated from the training dataset. Similarly to $I(\lambda)$ in Fig. 3.3, we observe a similar contrast pattern across different types of filters, as shown in Fig. 3.4. But $I(\lambda)$ displays a larger dynamic range than $\overline{\rho}(\lambda)$ due to its heavier penalty for large ρ.

We penalize filters with large ACC using the new objective function

$$e_j^{\text{REG}} = \sum_{t \in \mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t, y_t) \neq z_t\} + \gamma \overline{\rho}(h), \tag{3.22}$$

where $\overline{\rho}(h) = \overline{\rho}(\lambda)$ indicates the weak classifier h is parameterized by the filter λ. The ACC regularized (ACCR) Adaboost is given in Table 3.1 by replacing $R(h)$ with $\overline{\rho}(h)$.

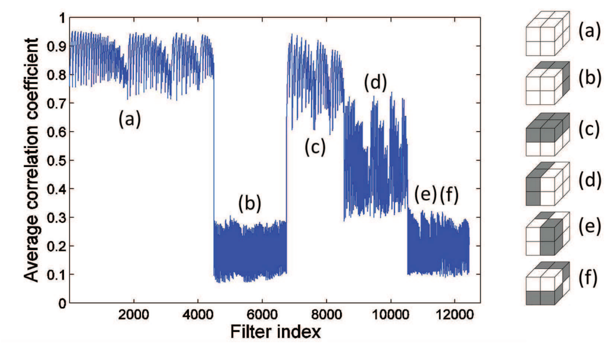


Figure 3.4: Average correlation coefficient $\overline{\rho}(\lambda)$ for the family of Haar-like filters on video slices.

3.5.3 Learning-Theoretic Analysis of the Regularized Adaboost Algorithm

In this section, we show that the regularized Adaboost in Table 3.1 fits an additive logistic regression model

$$\delta(x,y) = \sum_{1\leq j\leq J} \alpha_j h_j(x,y),$$

under the *regularized exponential loss* function

$$L(z,\delta(x,y)) \triangleq \exp\{-z\delta(x,y)\} + \gamma \sum_{1\leq j<J} 2\sinh(\alpha_j)R(h_j). \tag{3.23}$$

The analysis is inspired by [35, 36] and does not depend on the specific form of the regularizer. As long as the regularizer is a functional of h, it can be plugged into the regularized Adaboost algorithm and the same analysis applies, which makes this approach fairly general. Hence, we show the derivation for a generic regularizer $R(h)$.

Using the regularized exponential loss function (3.23), at iteration j, one must solve

$$(\alpha_j, h_j) = \arg\min_{\alpha\in\mathbb{R}, h\in\mathcal{H}} \left[\sum_{t\in\mathcal{T}} w_t^{(j)} \exp\{-\alpha z_t h(x_t,y_t)\} + 2\sinh(\alpha)\gamma R(h)\right], \tag{3.24}$$

where $w_t^{(j)} = \exp\{-z_t\delta_{j-1}(x_t,y_t)\}$ and

$$\delta_i(x,y) \triangleq \sum_{1\le j\le i} \alpha_j h_j(x,y).$$

Using the fact that $h(x,y) \in \{-1,1\}$, the objective function of (3.24) can be expressed as

$$\left[e^{-\alpha} \sum_{h(x_t,y_t)=z_t} w_t^{(j)} + e^{\alpha} \sum_{h(x_t,y_t)\neq z_t} w_t^{(j)}\right] + \left(e^{\alpha} - e^{-\alpha}\right)\gamma R(h),$$

which in turn can be written as

$$\left[\left(e^{\alpha} - e^{-\alpha}\right)\sum_{t\in\mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t,y_t)\neq z_t\} + e^{-\alpha}\sum_{t\in\mathcal{T}} w_t^{(j)}\right] + \left(e^{\alpha} - e^{-\alpha}\right)\gamma R(h).$$

Since $\sum_{t\in\mathcal{T}} w_t^{(j)} = 1$, the objective function becomes

$$2\sinh(\alpha)\left[\sum_{t\in\mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t,y_t)\neq z_t\} + \gamma R(h)\right] + e^{-\alpha}. \tag{3.25}$$

The minimum over $h \in \mathcal{H}$ is given by

$$h_j = \arg\min_{h\in\mathcal{H}} \sum_{t\in\mathcal{T}} w_t^{(j)} \mathbb{1}\{h(x_t,y_t)\neq z_t\} + \gamma R(h). \tag{3.26}$$

Plugging h_j into (3.25) and solving for α, we obtain

$$\alpha_j = \frac{1}{2}\log\frac{1-e_j^{\mathrm{REG}}}{e_j^{\mathrm{REG}}}, \tag{3.27}$$

where e_j^{REG} is given by (3.20). Equation (3.26) and (3.27) are equivalent to Step 1 and 2 of the regularized Adaboost algorithm in Table 3.1.

3.6 Experimental Results and Discussion

In this section, we test the proposed MIR and ACCR Adaboost algorithms for both video and audio content ID systems. The results are compared with SPB in [29] and [28]. We examine the content ID performance based on the two decoders defined in Section 3.3.

3.6.1 Video Fingerprinting

Experimental Setup

The video dataset we use contains 1,700 randomly selected videos from the publicly available Internet Archive videos (IACC.1.C) [37]. The archive covers a variety of genres including news, politics, animation, education, animals, vehicles, music and sports. We randomly divide the 1,700 videos into training, validation, and testing subsets consisting of 100, 100, and 1,500 videos respectively. From the training videos, we generate 20,000 matching and 20,000 nonmatching pairs ($|\mathcal{T}| = 40,000$) of 1 sec video sequences. The training pairs are generated from the following video distortions :

1. Cropping of 25%;
2. Resizing to CIF (352×288);
3. Frame rate change to 15 fps;
4. WMV lossy compression at 256 kb/s;
5. Rotation at 10 degrees;
6. Shifting downward and left by 20 pixels.

We adopt the same video normalization as in [29]. Videos are resampled at 10 fps, converted to grayscale, and resized to QVGA (320×240). These preprocessing steps aim to make the fingerprinting algorithm robust to frame rate change, color variation, and image resizing. After preprocessing, we extract intermediate features from each image before applying filters. The intermediate feature used in our experiments and in [29] is block mean luminance (BML), which is perceptually significant and reduces computational complexity. The BML is extracted on 36 (4×9) blocks per frame. One second of intermediate features ($4 \times 9 \times 10$ blocks) becomes the basic building block (a slice) for fingerprint extraction.

The training dataset contains an equal number of pairs from each distortion, and $J = 16$ filters and quantizers are selected by SPB or regularized Adaboost. Each filter output is quantized into 4 levels ($\mathcal{A} = \{a, b, c, d\}$) and converted to binary fingerprint by gray code. As noted in Section 2.2.2, SPB chooses each quantizer from 680 and 969 candidate quantizers for the video and audio fingerprinting systems respectively. Besides the high computation cost, these candidate quantizers are not chosen based on any optimality criterion for content ID performance. In the paper [27], where $|\mathcal{A}| = 2$, the authors noted that all thresholds learned by APB were approximately at the median of the filter response distribution. Putting a threshold at the median maximizes the bit entropy. Similarly, for $|\mathcal{A}|$-level quantization, we propose to use the $|\mathcal{A}|$ quantiles of the filter response distribution as the thresholds for a given filter. This quantization scheme produces bins with equal probabilities ($1/|\mathcal{A}|$) and therefore maximizes bit entropy. It also makes the training process much faster (nearly three orders of magnitude faster than evaluating 19 candidate thresholds). Note that achieving maximum entropy for each bit is a desirable property for many hashing algorithms [5, 12, 22, 38]. However, uniform distribution does not necessarily lead to maximization of the mutual information (except for some simple channels, e.g., symmetric channels), which is the proposed objective function for selecting hash functions in Section 3.4.1. Here, we choose such a quantization rule mainly due to the simpler training requirement.

We choose the regularization parameter λ by validation on a few candidate choices. The validation set contains 100 videos independent from both training and testing. For the single-output decoder, we select the λ that generates the smallest P_{FN} at a fixed P_{FP} of interest. While for the variable-size list decoder, we seek the smallest P_{miss} at a fixed $\mathbb{E}[N_{\mathrm{i}}]$ of interest.

The training time for regularized Adaboost is the same as SPB. To select 16 filters, the training time for both SPB and regularized Adaboost is 596 s on a desktop with Intel Xeon W3530 @ 2.80GHz processor and 6GB RAM.

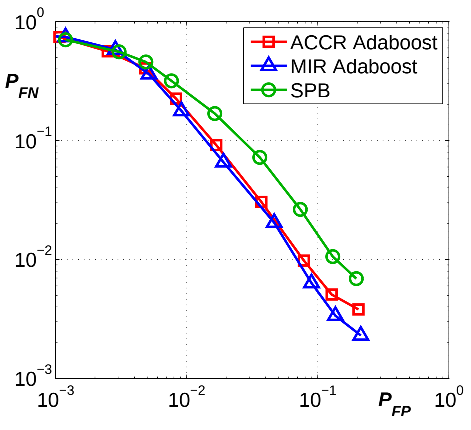
Selected Filters

In all our experiments, SPB selects more filters from the high-correlation group (type (a), (c) and (d) filters), whereas filters chosen by regularized Adaboost are dominated by the low-correlation group (type (b), (e) and (f) filters). As Adaboost reweighs training examples after each iteration, to correctly classify those higher weighted examples (incorrectly classified in previous iterations) may require a different type of filters. Thus, SPB selects different types of filters to best fit the training examples. However, in regularized Adaboost, reducing weighted classification error is not the only objective at each iteration. The ability to decorrelate overlapping slices in order to increase mutual information is also considered. The regularizers effectively demote filters of type (a), (c) and (d) which generate highly correlated responses on overlapping slices. The superiority of the low-correlation filters is demonstrated next in a comparative test.

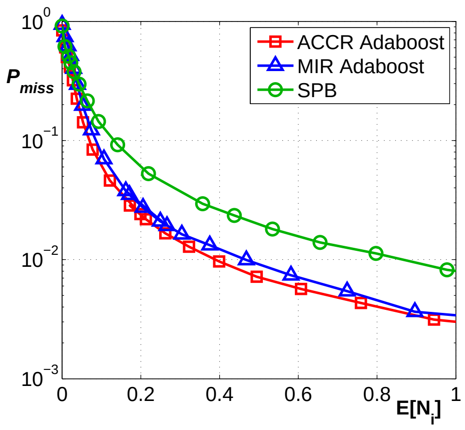
Comparative Test

To compare the content ID performance of SPB and regularized Adaboost, we generate $25,200$ queries of 10-second intermediate feature sequences by applying the six distortions to the 1,500 testing videos, where the original 1,500 testing videos serve as the database in estimating P_{FN}, P_{miss} and $\mathbb{E}[N_{\text{i}}]$. To estimate P_{FP} of the single-output decoder, we use the leave-one-subject-out (LOSO) scheme. In each run, we choose the samples from one testing video as the queries, and the remaining testing videos serve as the database.

Fig. 3.5 shows the content ID performance under the single-output decoder and the list decoder. Irrespective of the regularizer and decoder used, regularized Adaboost outperforms SPB. Fig. 3.5 reports the overall performance under different video distortions. The performance under each individual distortion follows the same trend. Note that $\mathbb{E}[N_{\text{i}}]$ can be much larger than one, but we only show $\mathbb{E}[N_{\text{i}}] < 1$ as they represent the most relevant region for a practical content ID system.



(a) Single-output decoder.



(b) List decoder.

Figure 3.5: Video content ID performance.

3.6.2 Audio Fingerprinting

Experimental Setup

The audio dataset is a collection of 1,700 songs spanning a variety of music genres including classical, vocal, rock and pop. We randomly divide the 1,700 songs into training, validation, and testing subsets consisting of 100, 100, and 1,500 songs respectively. From the training songs, we generate 22,400 matching and 22,400 nonmatching SSC image pairs. The audio distortions are created by the software GoldWave [39] and the audio distortions considered in this chapter are as follows:

1. Bandpass filtering (BPF): 400 Hz to 4 kHz bandpass filtering.
2. Echo (E): Tunnel reverberation.
3. Equalization 1 (EQ1): Boost bass.
4. Equalization 2 (EQ2): Recording industry association of America (RIAA).
5. Audio slice misalignment (ASM): 92.9 ms shift.
6. Sampling rate change (SR): Down-sampling to 16 kHz.
7. Volume change (V): Attack-Decay-Sustain-Release (ADSR) envelop.
8. WMA encoding (WMA): 64 kb/s WMA encoding.

On top of the above distortions, each audio signal is encoded by 96 kb/s MP3 encoding.

We follow the same experimental setup as in [28] for audio fingerprinting. An audio signal is first normalized to mono with 11,025 Hz sampling rate, and then converted into overlapping segments by a window with size 371.52 ms and shift 185.76 ms. For every segment, an M-dimensional spectral subband centroid (SSC) vector is computed [40] from $M = 16$ critical subband linearly spaced in mel scale from 300 Hz to 5300 Hz. A SSC image, built from $N = 10$ consecutive SSC vectors, is the basic building block (a slice) for fingerprint extraction. For every shift of 185.76 ms, an SSC image is obtained from an audio slice of length 2.04 s.

Different from the 3-D Haar-like filters for video (see Fig. 2.2), the candidate filters for audio are 2-D Haar-like filters (see Fig. 3.6) applied on $M \times N$ SSC images.

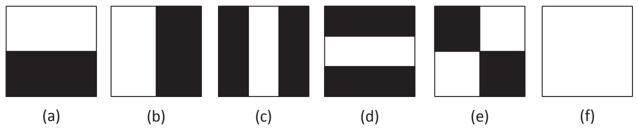
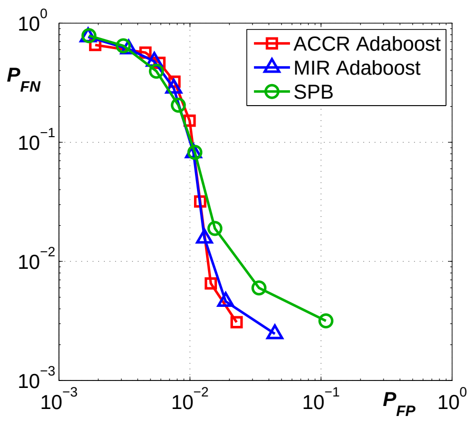
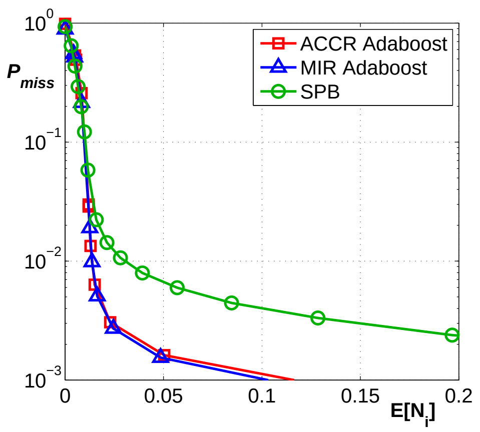


Figure 3.6: 2-D Haar-like filters [28]: The filter outputs are the average difference between values in light and dark regions.

The training set contains an equal number of SSC image pairs from each distortion. Similar to video fingerprinting, $J = 16$ filters are selected. Each filter response is quantized into $\mathcal{A} = 4$ levels and converted to binary fingerprint by gray code. The regularization parameter γ is chosen by validation on a few candidate choices. To select 16 filters, the training time for both SPB and regularized Adaboost is 152 s on a desktop with Intel Xeon W3530 @ 2.80GHz processor and 6GB RAM.



(a) Single-output decoder.



(b) List decoder.

Figure 3.7: Audio content ID performance.

Comparative Test

To compare the content ID performance of SPB and regularized Adaboost, we generate 112,000 queries of 10-second SSC images equally divided by the eight considered distortions. As in [28], the overlapping factor between two consecutive slices is 10/11 and a 10-second query corresponds to $L = 44$ slices.

Fig. 3.7 shows the audio content ID performance under both decoders. Irrespective of the regularizer and decoder used, regularized Adaboost outperforms SPB. Moreover, ACCR Adaboost and MIR Adaboost perform comparably.

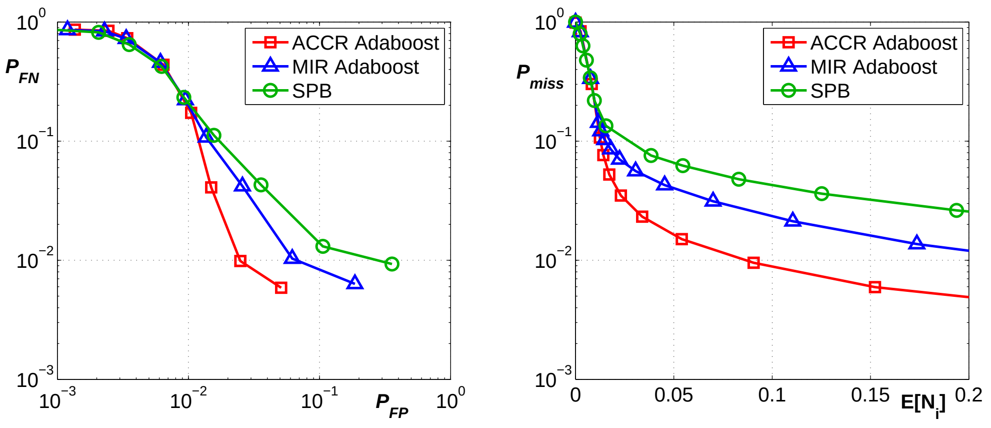
Audio Slice Misalignment

Most content ID systems use a high overlapping factor for fingerprint extraction. Though slice overlapping increases system complexity, it is a practical compromise to overcome misalignment between query fingerprint and database fingerprint. For our audio content ID system, a 2.04 s subfingerprint is extracted for every 185.76 ms shift. So the worst misalignment we will encounter is 92.9 ms, which represents less than 5% of a subfingerprint.

To further examine the effect on misalignment for audio content ID systems, we add the worst slice misalignment to each distortion. As shown in Fig. 3.8, MIR Adaboost and ACCR Adaboost still outperform SPB under both decoders. Comparing Fig. 3.7 with Fig. 3.8, both SPB and regularized Adaboost perform worse under slice misalignment. However, the misalignment has a stronger impact on MIR Adaboost than ACCR Adaboost because MIR penalizes filters generating highly correlated responses on two consecutive slices, where correlation between two consecutive slices helps alleviate slice misalignment. Note that the misalignment problem does not exist for our video query fingerprints because the shift is a single frame.

3.7 RGB-D Content Identification

Over the years, many image hashing algorithms have been proposed and have demonstrated good performance in large-scale similarity search applications [41]. Despite promising results, they still face the limitation that images are 2-D projections of the 3-D world and depth information is lost. Fortunately, advances in sensing technology have now made it possible to equip images with depth information. In particular, Xbox Kinect cameras are inexpensive and output both RGB and depth [42]. Kinect was originally designed for



(a) Single-output decoder. (b) List decoder.

Figure 3.8: Audio content ID performance with maximum misalignment.

gaming, but soon found applications to various research problems in Signal Processing, Computer Vision, Robotic Navigation, and Computer Graphics. The application of Kinect to real-time human pose recognition won the best paper at the top computer vision conference CVPR [43] in 2011. We expect that RGB+depth (RGB-D) data will become widespread in the future, and that databases such as [44, 45, 46, 47] will be commonplace. Since the combination of RGB and depth information is intrinsically more suitable than RGB alone or depth alone for representing scene content, the central goal of this chapter is to investigate how to combine RGB and depth to generate better hash codes for various similarity search applications such as RGB-D content ID and RGB-D near-duplicate image detection (NDID).

3.7.1 Kinect Depth Image

Depth information has traditionally been either estimated from RGB images using stereo matching, which is computationally expensive, or measured by expensive laser scanners. However, recent development in sensing technology makes depth acquisition computationally and financially more affordable. In particular, Xbox Kinect (see Fig. 3.9) outputs real-time, high-quality, synchronized videos of RGB and depth (RGB-D) at a cost of about one hundred dollars.

The Kinect sensor consists of an infrared laser emitter, an infrared camera,

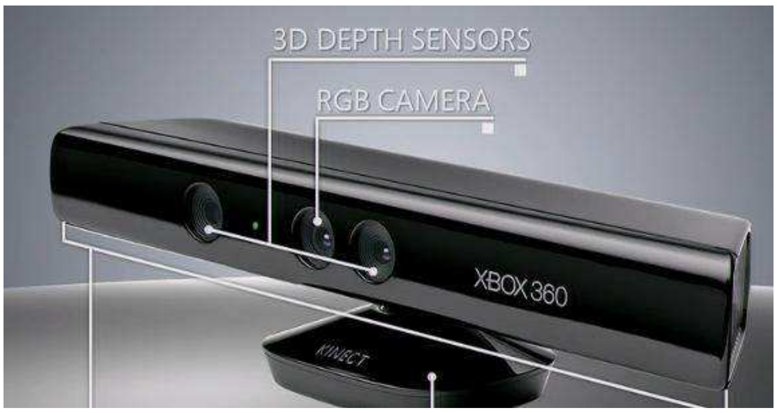


Figure 3.9: Xbox Kinect.

and an RGB camera, as shown in Fig. 3.9. The emitter emits fixed patterns to the environment, and the infrared camera receives the reflected signal. Then depth information is measured by a triangulation process [48]. The current Kinect works only in indoor environments and has a working range of 0.5 m to 5.0 m according to the specifications [48]. Still it has the potential to make consumer-grade video cameras produce RGB-D videos and project them in a 3D TV [49].

Following the introduction of Kinect, many datasets have been created and made publicly available including the *NYU depth V2 dataset* [44], *ADSC human daily activity dataset* [47], *LIRIS human activities dataset* [45], and *University of Washington RGB-D object dataset* [46]. Among them, the NYU depth V2 dataset, comprised of 464 scenes taken from three cities, captures the most comprehensive indoor environments and is used in all of our experiments.

Fig. 3.10 shows some random RGB-D images from the NYU depth V2 dataset. Depth images are not compressed, and each pixel is presented by the 11 bits outputted by Kinect. It is clear that depth images are quite noisy. There are missing pixel values which are caused by shadows from the disparity between the infrared emitter and camera or random missing or spurious values from specular or low albedo surfaces [44]. These missing values could be filled by a colorization scheme [44], but it is computationally expensive to fill one image, which makes it impractical for application demanding real-time search. Moreover, the intermediate feature uses only the block average information which alleviates the missing value problem.

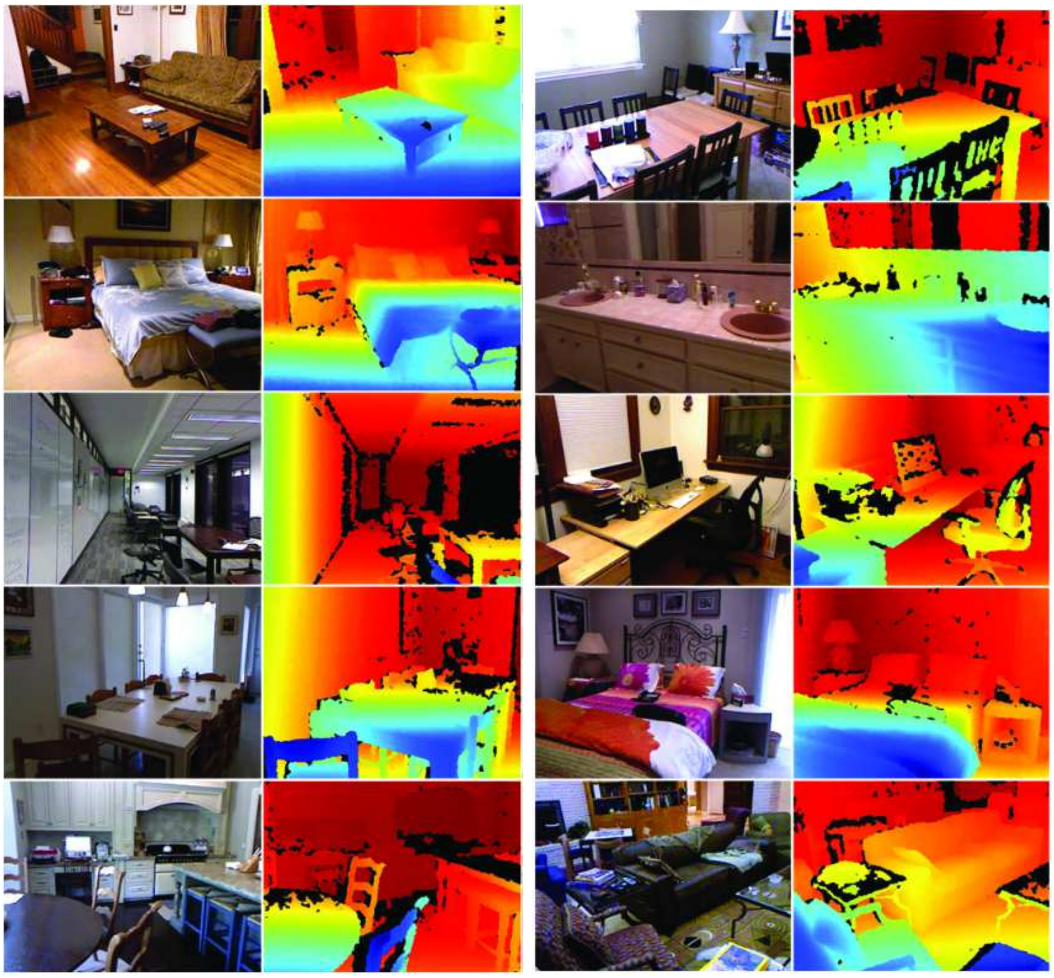


Figure 3.10: NYU depth dataset.

3.7.2 Statistical Difference between RGB and Depth Images

Intuitively, depth images contain more homogeneous patches and fewer localized features, such as lines, edges and corners. One way to quantify this is to fit the fine-scale wavelet coefficients into a two-parameter generalized Gaussian distribution (GGD) model [50, 51, 52]

$$P_X(x : s, p) = \frac{\exp\left(-|x/s|^p\right)}{Z(s, p)}, \tag{3.28}$$

where the normalization constant is $Z(s,p) = 2\frac{s}{p}\Gamma(\frac{1}{p})$ with Γ denoting the gamma function. Here, s is the standard deviation and p is the *shape parameter*. The GGD model contains the Gaussian and Laplacian probability density functions (PDFs) as special cases, using $p = 2$ and $p = 1$, respectively.

For decreasing values of p, the tails of the distribution become increasingly flat.

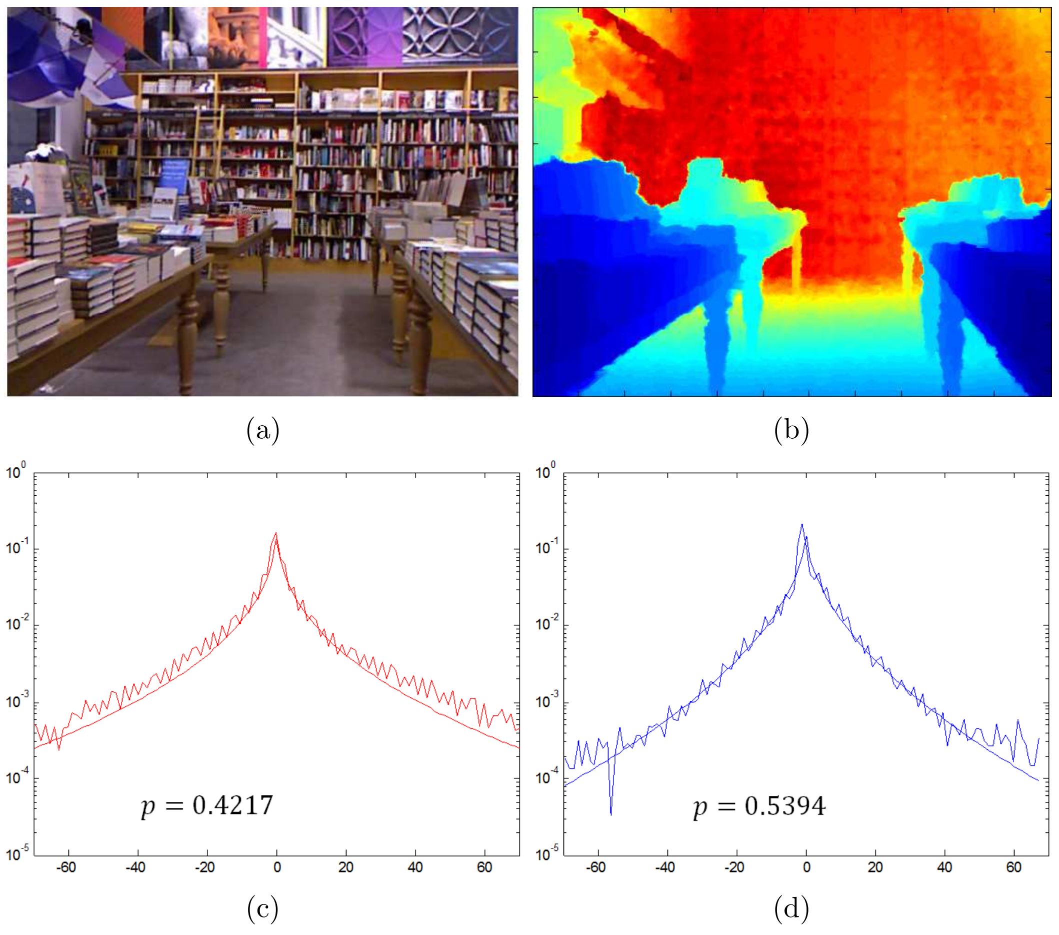


Figure 3.11: Log histogram of Haar wavelet coefficient in LH subband: (a) RGB image; (b) depth image; (c) GGD fit for RGB; (d) GGD fit for depth.

Fig. 3.11 shows the fit of the GGD model to the log histogram of Haar wavelet coefficients in the LH subband, which captures the image's horizontal edges. The GGD parameters are estimated by maximizing the likelihood of the data [53]. A larger shape parameter for the depth image suggests that the depth image contains fewer horizontal edges than the RGB image. This is confirmed by the absence of book edges in Fig. 3.11b.

Table 3.2: Statistics of the estimated shape parameters.

	Range	Mean
RGB	[0.15, 0.76]	0.4045
Depth	[0.16, 1.27]	0.6008

To be statistically significant, we fit the GGD model to another 1400 RGB+depth image pairs and summarize the estimated shape parameters in Table 3.2. The large difference between mean values of the shape parameters is consistent with our intuition that depth images contain more homogeneous regions and fewer localized features.

3.7.3 RGB-D Content ID Systems

Depth Features

As Fig. 3.11 illustrates, depth images contain more homogeneous patches and fewer localized features, such as lines, edges and corners, than RGB images. If the intermediate feature $x \in \mathcal{X}$ consists of averages of homogeneous spatiotemporal patches of a depth video segment, x is approximately a sufficient statistic for the depth video segment. In practice, determining the number of homogeneous patches and their locations can be difficult and time consuming, and thus we propose the block mean depth (BMD) as intermediate features for depth video. First, each depth frame is divided into $N_r \times N_c$ blocks (N_r rows and N_c columns). The intermediate feature x at block $B_{r,c,t}$ in the r-th row, c-th column and t-th ($1 \leq t \leq T$) frame of a depth video segment is calculated as

$$x(r,c,t) = \frac{1}{|B_{r,c,t}|} \sum_{(i,j) \in B_{r,c,t}} d(i,j,t), \tag{3.29}$$

where $|\cdot|$ denotes set cardinality, and $d(i,j,t)$ is the depth value at coordinates (i,j) in the t-th frame. Hence, the feature space $\mathcal{X} = \mathbb{R}^{N_r \times N_c \times T}$. The averaging operation in the BMD feature makes it relatively robust to the acquisition noise in depth frames. We use the BML as the intermediate feature for RGB videos.

The Hybrid System

As illustrated in Fig. 3.11, most humans can reasonably infer the depth information from the corresponding RGB image. A mathematical algorithm has recently been developed to estimate depth information from a single RGB image [54]. However, this kind of inference requires global image processing, and local features such as block mean luminance (for RBG) and block mean depth are more likely to be independent. Thus we can build a hybrid system to harvest the diversity gain, when both RGB and depth are available in video signals.

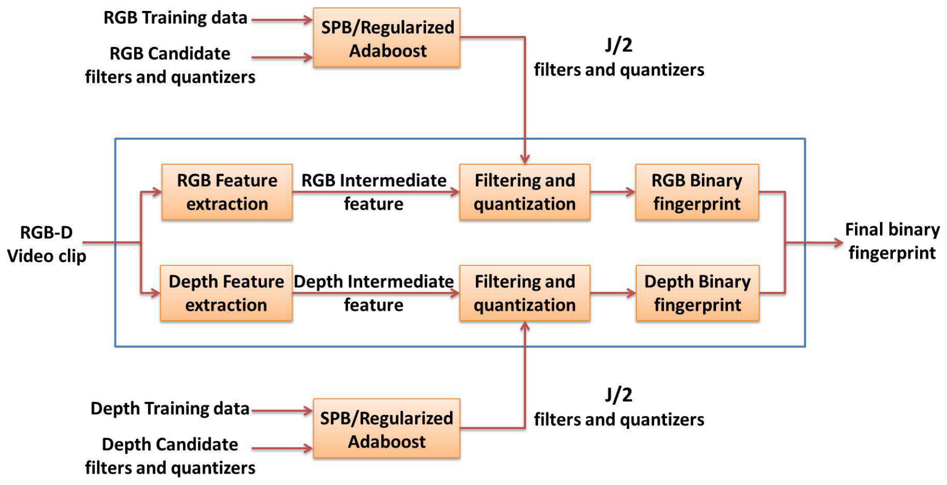


Figure 3.12: Fingerprint extraction for a hybrid system.

Fig. 3.12 illustrates our RGB-D fingerprint code design. We train half of the filters and quantizers from RGB intermediate features, and the other half from depth intermediate features. Thus for each RGB-D video, half of the fingerprint is generated from RGB and half from depth. Then filtering and quantization are applied, and the combined final fingerprint is used to identify the video in the hybrid system.

Performance Evaluation

Experimental Setup: The NYU Depth V2 dataset captures comprehensive indoor environments and used in our experiments. It is comprised of 464 indoor scenes taken from three cities. Each scene is recorded as a short RGB-D video. We exclude extremely short videos as well as videos with frame rate less than 15 fps from our experiments. The 380 videos left are randomly divided into a training set of 100 videos and a testing set of 280 videos. From the training data, we generate 16,000 matching and 16,000 nonmatching pairs ($|\mathcal{T}| = 32,000$) of sequences of intermediate features from 10 consecutive synchronized RGB and depth frames. The training pairs are generated from the video distortions illustrated in Fig. 3.13: 50% cropping, vertical mirroring, frame rotation of 15 degrees and frame shifting downward and left by 100 pixels. We consider geometric distortions only as they represent the most challenging video distortions to detect.

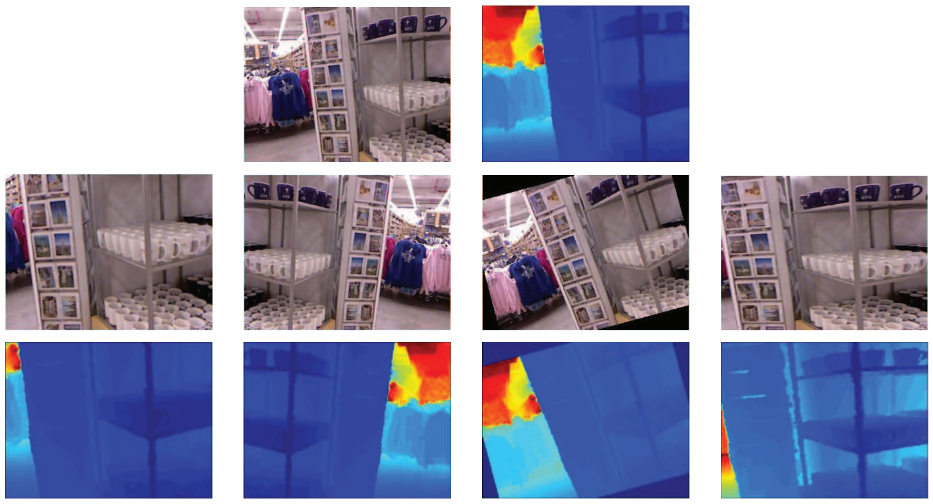


Figure 3.13: Sample distorted images. Top row: Original RGB and depth images. Bottom two rows: Distorted RGB and depth images. Distortions from left to right are: cropping of 50%, vertical mirroring, rotation of 15 degree and shift downward and left by 100 pixels.

We adopt the same video normalization as in [29]. Before extraction of intermediate features, both RGB and depth videos are resampled at 10 frames per second, converted to grayscale (RGB only) and resized to QVGA (320x240). These preprocessing steps aim to make the fingerprinting algorithm robust to frame rate change, color variation, and frame resizing. After preprocessing, block mean luminance (BML) and block mean depth (BMD) are extracted from RGB and depth video clips on 36 ($N_r = 4, N_c = 9$) blocks per frame. The temporal length of the intermediate features is 1 second, and the query length is 5 seconds with overlapping factor of 9/10. We train $J = 16$ classifiers each for RGB and depth. The first 8 classifiers from RGB and depth are combined to generate hybrid fingerprints. Each filter output is quantized into 4 levels. Hence our query fingerprint is 1312 bits long. For regularized Adaboost, we use MIR Adaboost with a regularization parameter of $\gamma = 0.03$ for both RGB and depth.

Experimental Results: The first result we notice from Fig. 3.14 is that the hybrid system outperforms the RGB-alone and depth-alone systems for all the considered distortions, irrespective of the fingerprinting algorithms used. We expect hybrid systems to perform better, but the gain is rather significant, often in orders of magnitude. In Fig. 3.14d, the hybrid system even presents a perfect ROC curve for the 20,500 queries in our experiment.

Secondly, depth systems outperform RGB systems uniformly. This is surprising because depth is often considered as a supplement to RGB. One possible explanation is that the rich information of RGB is compressed heavily by the black mean features, and it gives an unfair advantage to depth. Experiments with other intermediate features are on the way.

Lastly, regularized Adaboost performs significantly better than SPB, irrespective of the modalities used. This is just another testimony to the superior performance of the regularized Adaboost algorithm, in addition to the video and audio results in Section 3.6.

Statistical Interpretation: In general, the superiority of regularized Adaboost over SPB stems from its ability to select more independent features. A similar phenomenon applies here, especially when we consider RGB features and depth features.

We first define the within-modality correlation of two filters λ_j and λ_k $(1 \leq j, k \leq J)$ as

$$R^m(j,k) = \frac{\mathbb{E}\left[(\lambda_j^m(X^m) - \mu_j^m)(\lambda_k^m(X^m) - \mu_k^m)\right]}{\sigma_j^m \sigma_k^m}, \tag{3.30}$$

where $m \in \{\text{RGB, D}\}$ denotes RGB and depth (D) respectively, X^m is the intermediate feature of one segment from the corresponding modality, μ_j^m and σ_j^m are the mean and standard deviation of $\lambda_j^m(X^m)$. We also define the between-modality correlation

$$R^{\text{RGB-D}}(j,k) = \frac{\mathbb{E}\left[(\lambda_j^{\text{RGB}}(X^{\text{RGB}}) - \mu_j^{\text{RGB}})(\lambda_k^{\text{D}}(X^{\text{D}}) - \mu_k^{\text{D}})\right]}{\sigma_j^{\text{RGB}} \sigma_k^{\text{D}}}, \tag{3.31}$$

the average absolute within-modality correlation,

$$\overline{R}^m = \frac{2}{J^2 - J} \sum_{j=1}^{J-1} \sum_{k=j+1}^{J} |R^m(j,k)|, \tag{3.32}$$

and the average absolute between-modality correlation,

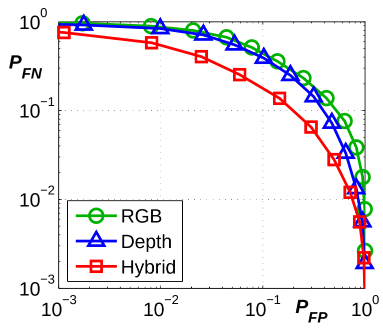
$$\overline{R}^{\text{RGB-D}} = \frac{1}{J^2} \sum_{j=1}^{J} \sum_{k=1}^{J} |R^{\text{RGB-D}}(j,k)|, \tag{3.33}$$

of these filters. The expectations are estimated from the training dataset, and their values are shown in Table 3.3. The average between-modality correlation is almost an order of magnitude smaller than the average within-modality correlation.

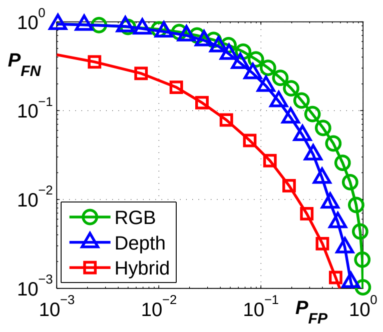
Table 3.3: Average within-modality and between-modality correlations.

	$\overline{R}^{\text{RGB}}$	$\overline{R}^{\text{D}}$	$\overline{R}^{\text{RGB-D}}$
SPB	0.1496	0.1025	0.0208
Regularized Adaboost	0.2303	0.1705	0.0199

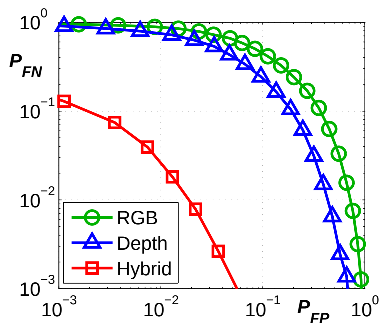
We also show in Fig. 3.15 the distributions of Hamming distance for matching and nonmatching pairs under the vertical mirroring distortion (other distortions exhibit the same trend). The better histogram separation of the second row is consistent with regularized Adaboost's superior content ID performance in Fig. 3.14d over SPB in Fig. 3.14c. The clear improvement in histogram separation from the left two columns (single modality) to the last column (RGB-D) is consistent with the better ROC curves of hybrid systems in Fig. 3.14c and Fig. 3.14d. Overall, a hybrid system based on regularized Adaboost performs significantly better than the other systems we considered.



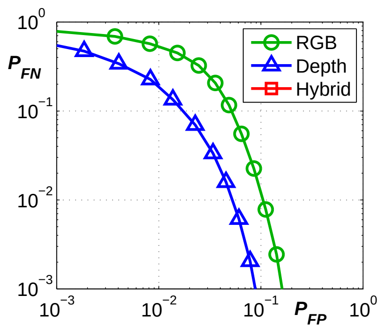
(a) Cropping



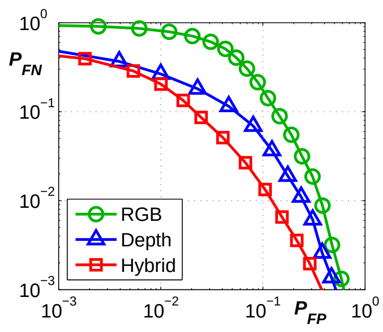
(b) Cropping



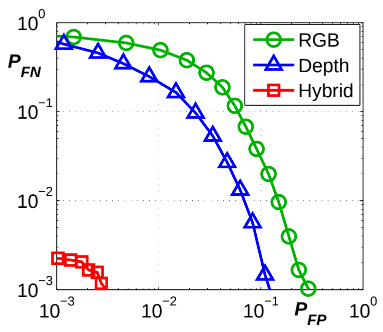
(c) Vertical mirroring



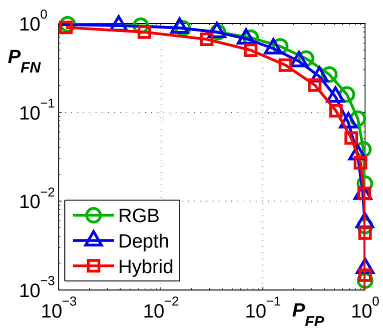
(d) Vertical mirroring



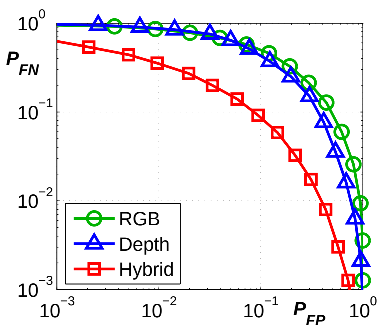
(e) Rotation



(f) Rotation



(g) Shift



(h) Shift

Figure 3.14: ROC curves for SPB (1st column) and regularized Adaboost (2nd column) under various distortions.

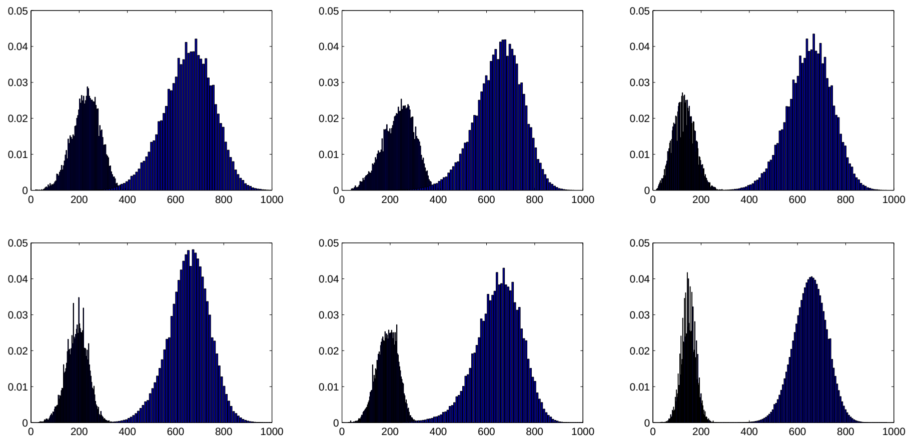


Figure 3.15: Distributions of Hamming distance for matching and nonmatching pairs for the vertical mirroring distortion. First row from left to right are SPB for: depth, RGB, and RGB-D. Second row from left to right are regularized Adaboost for: depth, RGB, and RGB-D.

CHAPTER 4

SNR MAXIMIZATION HASHING

A popular family of hash functions, which assumes centered (mean-subtracted) inputs $\mathbf{x} \in \mathbb{R}^d$, a projection matrix $W \in \mathbb{R}^{d\times k}$, and binary scalar quantization, is given by

$$h(\mathbf{x}, W) = \operatorname{sgn}(W^T\mathbf{x}) \in \{\pm 1\}^k, \tag{4.1}$$

where $\operatorname{sgn}(v) = 1$ if $v \geq 0$ and -1 otherwise. For a matrix or vector, $\operatorname{sgn}(\cdot)$ denotes the element-wise operation. Many hashing algorithms fall in this category [2, 12, 22, 6, 55]. Other families of hash functions based on learning kernels [56], multilayer neural networks [11, 7], and boosting [28, 57] are more expensive to train and evaluate.

Traditionally, W was generated by randomly sampling a distribution that satisfies the locality-sensitive property [20, 4]. However, data-independent W can lead to inefficient codes, and thus require much longer codes (larger k) to work well. Recently, learning W from training datasets has been shown to outperform data-independent W for the same code length [12, 6, 7, 55]. To learn W, the nondifferentiable and nonconvex sgn function of (4.1) is often approximated by either the identity function $h(\mathbf{x}; W) = W^T\mathbf{x}$ [12, 22, 6], which obviously introduces a large approximation error when the magnitude of $W^T\mathbf{x}$ is large, or the hyperbolic tangent function $\tanh(W^T\mathbf{x})$ [55], which can cause the optimization to be trapped in a bad local optimum due to the nonconvexity of the tanh function.

In this chapter, we introduce SNR maximization as a candidate for selecting the projection directions. We show that this approach minimizes the hashing error probability under a Gaussian model. We propose a SNR maximization hashing (SNR-MH) algorithm that iteratively finds uncorrelated projections that maximize the SNR. Our method does not require any approximation to the sgn function and finds the global optimal solution.

SNR has been used as the performance measure in many applications, such as lossy compression [58], matched filtering [59], relay functionality in memoryless relay networks [60], and beamforming in narrowband sensor arrays [61, 62]. Among these applications, matched filtering and beamforming are closely related to our linear projection learning. In both matched filtering and beamforming, the observed signal $\mathbf{Y} = \mathbf{X}+\mathbf{Z} \in \mathbb{R}^d$ consists of the desired signal $\mathbf{X} \in \mathbb{R}^d$ corrupted by independent additive noise $\mathbf{Z} \in \mathbb{R}^d$.

In matched filtering, the signal $\mathbf{X}$ is deterministic and consists of temporal samples. In beamforming, the signal $\mathbf{X}$ is stochastic and consists of samples from spatially separated sensors. In both cases, the goal is to construct a linear filter $w^* \in \mathbb{R}^d$ such that the signal-to-noise ratio at the filter output is maximized:

$$w^* = \arg\max_w \ \frac{w^T R_X w}{w^T R_Z w}, \tag{4.2}$$

where $R_Z \triangleq \mathbb{E}[\mathbf{Z}\mathbf{Z}^T]$ is the noise auto-correlation matrix and $R_X \triangleq \mathbb{E}[\mathbf{X}\mathbf{X}^T]$ is the signal autocorrelation matrix. Note that R_X reduces to $R_X = \mathbf{X}\mathbf{X}^T$ when $\mathbf{X}$ is deterministic. The solution to (4.2) can be obtained by solving a generalized eigenproblem. To guarantee uniqueness of the solution, the noise power is usually normalized, e.g., $w^T R_Z w = 1$. For matched filtering, when $\mathbf{Z}$ is additive white Gaussian noise (AWGN) with covariance matrix $R_Z = \sigma_Z^2 I$, the solution w^* is a scaled version of $\mathbf{X}$, i.e., $w^* = c\mathbf{X}$, which not only maximizes SNR, but can be used for optimal detection as well. For beamforming, w^* is the optimal transformation that linearly combines d different copies of the desired signal from d sensors.

Unlike matched filtering and beamforming, hashing does not aim to recover $\mathbf{X}$. Rather, the decision to be made is whether two signals $\mathbf{X}$ and $\mathbf{Y}$ are related or not. The decision is not based on $\mathbf{X}$ and $\mathbf{Y}$ directly but on binary hash codes extracted from $\mathbf{X}$ and $\mathbf{Y}$. Moreover, hashing learns k projection vectors $\{w_1, \ldots, w_k\}$ instead of just one vector as in the matched filtering and beamforming applications.

To our knowledge, SNR has not yet been used as the performance measure for hashing in the literature. In the next section, we show that maximizing SNR is equivalent to minimizing the hashing error probability in a Gaussian model. In Section 4.2, we derive the SNR-MH algorithm and relate it to other hashing algorithms. Section 4.3 contains simulation results for both synthetic and real datasets, demonstrating SNR-MH's superior performance in learning compact binary codes.

4.1 Statistical Model

In this section, we introduce a second-order statistical model for hashing and motivate SNR maximization by showing that under an additional Gaussian assumption, a larger SNR results in a smaller hashing error probability.

4.1.1 Statistical Model for Hashing

The second-order statistical model consists of the following ingredients:

(A1) The signal $\mathbf{X} \in \mathbb{R}^d$ follows a distribution $P_{\mathbf{X}}$ with mean $\mathbf{0}$ and covariance matrix $C_X \in \mathbb{R}^{d\times d}$.

(A2) If the query item $\mathbf{Y}$ is related to $\mathbf{X}$, the following distortion model holds:

$$\mathbf{Y} = \mathbf{X} + \mathbf{Z}, \tag{4.3}$$

where the noise $\mathbf{Z}$ is independent of $\mathbf{X}$ and follows a distribution $P_{\mathbf{Z}}$ with mean $\mathbf{0}$ and positive-definite covariance matrix C_Z.

(A3) If $\mathbf{X}$ and $\mathbf{Y}$ are unrelated, $\mathbf{Y}$ is independent of $\mathbf{X}$ and follows a distribution $P_{\mathbf{Y}}$.

The hashing code is as follows:

(A4) The projection matrix $W \in \mathbb{R}^{d\times k}$, $k \le d$ is such that the $k \times k$ matrices $W^T C_X W$ and $W^T C_Z W$ are both diagonal.[1] Hence, the transformed feature components $\{w_i^T \mathbf{X}\}_{i=1}^k$ are uncorrelated, and the transformed noise components $\{w_i^T \mathbf{Z}\}_{i=1}^k$ are also uncorrelated. Denote by $\{\sigma_i^2\}_{i=1}^d$ and $\{\lambda_i^2\}_{i=1}^d$ the diagonal entries of $W^T C_X W$ and $W^T C_Z W$ respectively. For the i-th projection, we have $w_i^T \mathbf{Y} = w_i^T \mathbf{X} + w_i^T \mathbf{Z}$. We define the i-th signal-to-noise ratio for the i-th projection as

$$\mathrm{SNR}_i \triangleq \frac{\sigma_i^2}{\lambda_i^2}, \quad 1 \le i \le k. \tag{4.4}$$

[1]The existence of such W is guaranteed [63, Theorem 15.3.2].

(A5) Binary fingerprints are extracted using the component-wise sgn function:

$$\begin{aligned}\mathbf{F} &= \operatorname{sgn}(W^T\mathbf{X}) \in \{\pm 1\}^k \\ \mathbf{G} &= \operatorname{sgn}(W^T\mathbf{Y}) \in \{\pm 1\}^k,\end{aligned} \tag{4.5}$$

with $F_i = \operatorname{sgn}(w_i^T\mathbf{X})$ and $G_i = \operatorname{sgn}(w_i^T\mathbf{Y}), 1 \le i \le k$.

(A6) Upon seeing a pair $(\mathbf{x}, \mathbf{y})$, a binary decision about whether $\mathbf{x}$ and $\mathbf{y}$ are *similar* or *dissimilar* is made based on the fingerprints $\mathbf{f} = \operatorname{sgn}(W^T\mathbf{x})$ and $\mathbf{g} = \operatorname{sgn}(W^T\mathbf{y})$. Similar and dissimilar $(\mathbf{x}, \mathbf{y})$ pairs are defined as follows:

$$\begin{aligned}\text{Similar } (S) &: \mathbf{x} \text{ and } \mathbf{y} \text{ are related by (4.3);} \\ \text{Dissimilar } (D) &: \mathbf{x} \text{ and } \mathbf{y} \text{ are independent.}\end{aligned}$$

(A7) The decision rule is

$$d_H(\mathbf{f}, \mathbf{g}) \underset{D}{\overset{S}{\lesseqgtr}} \tau, \tag{4.6}$$

where $d_H(\mathbf{f}, \mathbf{g}) \triangleq \sum_{i=1}^k 1_{\{f_i \neq g_i\}}$ is the Hamming distance between $\mathbf{f}$ and $\mathbf{g}$, and $\tau \in \{0, 1, \ldots, k\}$ is a decision threshold. The rule declares $(\mathbf{x}, \mathbf{y})$ similar when $d_H(\mathbf{f}, \mathbf{G}) \le \tau$ and dissimilar when $d_H(\mathbf{f}, \mathbf{G}) > \tau$. A refinement on (4.6) would be to randomize the decision in the event that $d_H(\mathbf{f}, \mathbf{g}) = \tau$. This make it possible to achieve a desired false-positive error probability, as will be discussed in Sections 4.3.1 and 5.2.

These assumptions on the hashing system are motivated by practical designs such as in [5, 2, 12, 22, 6, 55]. In particular, the uncorrelatedness property of hash codes in (A4) was first proposed in [5] and used by many subsequent hashing algorithms [12, 64, 65]; hash functions in the form of (A5) were used in [2, 12, 22, 6, 55], and the decision rule (A7) is widely used as Hamming distance can be computed extremely fast using bitwise XOR.

4.1.2 Error Probability Analysis under Gaussian Model

Based on the above statistical model, we analyze the hashing error probabilities under the additional assumption that $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, C_X)$ and $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, C_Z)$.

For Gaussian random vectors, uncorrelatedness of $W^T\mathbf{X}$ and $W^T\mathbf{Z}$ implies independence. It then follows from (A4) that $\{F_i\}_{i=1}^k$ are independent and from (A2) that so are $\{G_i\}_{i=1}^k$. For non-Gaussian $\mathbf{X}$ and $\mathbf{Y}$, we would only have uncorrelated $\{F_i\}_{i=1}^k$ and uncorrelated $\{G_i\}_{i=1}^k$.

Denote by $P_{\mathbf{FG}}(\mathbf{f}, \mathbf{g}) = \prod_{i=1}^k P_{F_iG_i}(f_i, g_i)$ the joint distribution of $(\mathbf{F}, \mathbf{G})$ when $\mathbf{X}$ and $\mathbf{Y}$ are similar and by $P_{\mathbf{F}}P_{\mathbf{G}}(\mathbf{f}, \mathbf{g}) = \prod_{i=1}^k P_{F_i}P_{G_i}(f_i, g_i)$ the distribution when $\mathbf{X}$ and $\mathbf{Y}$ are dissimilar. The performance of the hashing system is quantified using probability of miss

$$P_M \triangleq P_{\mathbf{FG}}\{d_H(\mathbf{F}, \mathbf{G}) > \tau\} \tag{4.7}$$

and probability of false alarm

$$P_F \triangleq P_{\mathbf{F}}P_{\mathbf{G}}\{d_H(\mathbf{F}, \mathbf{G}) \le \tau\}. \tag{4.8}$$

In the rest of this section, we prove the following proposition with the help of Lemmma 1 and 2 below.

Proposition 1. *Under the Gaussian model, for a fixed τ, P_M is a decreasing function of $\{\mathrm{SNR}_i\}_{i=1}^k$ and P_F is independent of $\{\mathrm{SNR}_i\}_{i=1}^k$.*

Proof. When $\mathbf{F} = \mathrm{sgn}(W^T\mathbf{X})$ and $\mathbf{G} = \mathrm{sgn}(W^T\mathbf{Y})$ are generated from independent $\mathbf{X}$ and $\mathbf{Y}$, we have

$$P_{F_i}P_{G_i}\{F_i \ne G_i\} = \frac{1}{2}, \; 1 \le i \le k. \tag{4.9}$$

As the pairs $(F_i, G_i), 1 \le i \le k$ are independent, $d_H(\mathbf{F}, \mathbf{G})$ follows the binomial distribution with k trials and parameter $\frac{1}{2}$:

$$d_H(\mathbf{F}, \mathbf{G}) \sim \mathrm{Bi}(k, \frac{1}{2}). \tag{4.10}$$

Hence, P_F does not depend on $\{\mathrm{SNR}_i\}_{i=1}^k$.

When $\mathbf{F}$ and $\mathbf{G}$ are generated from similar $\mathbf{X}$ and $\mathbf{Y}$, define

$$p_i \triangleq P_{F_iG_i}\{F_i \ne G_i\}, \quad 1 \le i \le k. \tag{4.11}$$

Since the pairs $(F_i, G_i), 1 \le i \le k$ are independent, the Hamming distance between $\mathbf{F}$ and $\mathbf{G}$ follows the Poisson binomial distribution (PBD) with parameter $\{p_1, \ldots, p_k\} \in [0,1]^k$:

$$P_{\mathbf{FG}}\{d_H(\mathbf{F}, \mathbf{G}) = l\} = \sum_{A \in E_l} \prod_{i \in A} p_i \prod_{j \in A^c} (1 - p_j), \quad 0 \le l \le k, \tag{4.12}$$

where E_l is the set of all subsets of l integers that can be selected from $\{1, 2, \ldots, k\}$ and $A^c = \{1, 2, \ldots, k\} \backslash A$ is the complement of A. In the special case of uniform probabilities $p_i \equiv p$, we have $d_H(\mathbf{F}, \mathbf{G}) \sim \mathrm{Bi}(k, p)$.

Define the random variable $T_k^S = d_H(\mathbf{F}, \mathbf{G})$ for similar $\mathbf{F}$ and $\mathbf{G}$, so $T_k^S \sim \mathrm{PBD}(\{p_1, \ldots, p_k\})$. Then we have $P_M = \Pr\{T_k^S > \tau\}$.

Lemma 1. *For a given decision threshold $\tau \in \{0, 1, \ldots, k-1\}$ and probabilities $\{p_1, p_2, \ldots, p_{k-1}\}$, $\Pr\{T_k^S > \tau\}$ is an increasing function of p_k.*

Proof. Let $T_{k-1}^S \sim \mathrm{PBD}(\{p_1, \ldots, p_{k-1}\})$. For $l = 0, 1, \ldots, k$, we have

$$\Pr\{T_k^S = l\} = p_k \times \Pr\{T_{k-1}^S = l - 1\} + (1 - p_k) \times \Pr\{T_{k-1}^S = l\}. \tag{4.13}$$

Since every PBD is unimodal, and the mode is either unique or shared by two adjacent integers [66], let l^* be the unique mode (or the smaller of the two modes) of T_{k-1}^S. When $l \le l^*$, we have $\Pr\{T_{k-1}^S = l-1\} < \Pr\{T_{k-1}^S = l\}$, so $\Pr\{T_k^S = l\}$ decreases with p_k. When $l > l^*$ (or $l > l^* + 1$ when there are two modes), we have $\Pr\{T_{k-1}^S = l - 1\} > \Pr\{T_{k-1}^S = l\}$, so $\Pr\{T_k^S = l\}$ increases with p_k.

Therefore, when $0 \le \tau \le l^*$, $\Pr\{T_k^S > \tau\} = 1 - \sum_{l=0}^{\tau} \Pr\{T_k^S = l\}$ is an increasing function of p_k. When $l^* + 1 \le \tau \le k - 1$, $\Pr\{T_k^S > \tau\} = \sum_{l=\tau+1}^{k} \Pr\{T_k^S = l\}$ is also an increasing function of p_k. □

Lemma 2. *Under (A2) and (A4), p_i is a decreasing function of SNR_i for $i = 1, \ldots, k$.*

Proof. Denote by $\widetilde{X}_i = w_i^T \mathbf{X}$ and $\widetilde{Z}_i = w_i^T \mathbf{Z}$ the i-th transformed feature random variable and transformed noise random variable respectively. Then $\widetilde{X}_i \sim \mathcal{N}(0, \sigma_i^2)$ and $\widetilde{Z}_i \sim \mathcal{N}(0, \lambda_i^2)$. By (4.3) and (4.5), $F_i = \mathrm{sgn}(\widetilde{X}_i)$ and $G_i = \mathrm{sgn}(\widetilde{X}_i + \widetilde{Z}_i)$ are independent. It has been shown in [67, Equations 16 and 17] that

$$p_i = P_{F_i G_i}\{F_i \ne G_i\} = \frac{1}{\pi} \arctan\left(\frac{1}{\mathrm{SNR}_i}\right), \tag{4.14}$$

which is a decreasing function of SNR_i. $\square$

It follows from (4.7), Lemma 1, and Lemma 2 that for fixed τ, P_M is a decreasing function of $\{\text{SNR}_i\}_{i=1}^k$. $\square$

4.2 SNR Maximization Hashing

Motivated by Proposition 1, we propose SNR-MH, a hashing algorithm that finds the globally optimal projection directions $\{w_i\}_{i=1}^k$ and then extracts binary fingerprints according to (4.5).

Denote by $\mathbf{X} \in \mathbb{R}^d$ and $\mathbf{Z} \in \mathbb{R}^d$ the feature random vector and noise vector respectively (both have mean zero). The problem is to find a $d \times k$ transformation matrix $W = [w_1, \ldots, w_k]$ such that the transformed feature vector $W^T\mathbf{X} \in \mathbb{R}^k$ is uncorrelated and the SNR at each projection $\text{SNR}_i = \text{var}(w_i^T\mathbf{X})/\text{var}(w_i^T\mathbf{Z})$ is maximized. Mathematically, the projection vectors w_i, $i = 1, 2, \ldots, k$, are learned sequentially via the following optimization:

$$\begin{aligned} w_i = \arg\max_{w} \quad & \frac{w^T C_X w}{w^T C_Z w} \\ \text{subject to} \quad & w^T C_X w_j = 0, \quad \forall j < i \\ & w^T C_Z w_j = 0, \quad \forall j < i \\ & w^T C_Z w = 1, \end{aligned} \tag{4.15}$$

where C_X and C_Z are the covariance matrices of $\mathbf{X}$ and $\mathbf{Z}$ respectively, and the last constraint is to normalize the transformed noise to unit power so the solution is unique. To ensure C_Z is invertible, a small constant $\alpha > 0$ is often added to the diagonal entries of C_Z, i.e., C_Z is replaced with $C_Z + \alpha I$ where I denotes the identity matrix.

The optimization (4.15) is used in multiclass Fisher discriminant analysis (FDA) [68] to learn up to k linear projections when there are $k+1$ different classes. In multiclass FDA, C_X is the inter-class scatter matrix and C_Z is the intra-class scatter matrix. The solution of (4.15) is given by the k eigenvectors corresponding to the first k largest eigenvalues of the generalized eigenproblem [68] (proof is given in Appendix B.)

$$C_X w = \gamma C_Z w, \tag{4.16}$$

where γ is the eigenvalue (to be interpreted as the SNR in the direction w).

There are several ways to reduce (4.16) to a standard eigendecomposition problem [63]. One way is to form $C_Z^{-1}C_X$, but in general $C_Z^{-1}C_X$ is not symmetric, so all the nice properties about diagonalizing symmetric matrices will be lost.

Another way to solve (4.16) is by using the Cholesky decomposition on C_Z [63]. Let $C_Z = LL^T$ where L is a lower triangular matrix. Then (4.16) becomes

$$\left[L^{-1}C_X L^{-T}\right]\left[L^T w\right] = \gamma\left[L^T w\right], \tag{4.17}$$

which is a standard eigendecomposition problem.

Note that the above procedure is equivalent to applying a whitening transformation L^{-1} on the noise. After whitening, $L^{-1}\mathbf{Z}$ and $L^{-1}\mathbf{X}$ have covariance matrices $L^{-1}C_Z L^{-T} = I$ and $L^{-1}C_X L^{-T}$ respectively.

Connection to PCA Hashing (PCAH): In PCA hashing [12, 6], W is given by the top k eigenvectors of C_X. This is equivalent to assuming C_Z is the identity matrix in (4.16). PCA hashing maximizes the transformed feature variance without considering the noise. The only case PCA hashing is optimal in the sense of SNR maximization is when the noise $\mathbf{Z}$ has uncorrelated components with equal variances.

Connection to Semi-Supervised Hashing (SSH): SSH [12] was formulated as maximizing a measure of classification accuracy while having large variance and quasi-independence of the hash bits. After approximating the sgn function with the identity function, SSH maximizes the following objective function subject to the constraint $W^T W = I$:

$$\sum_{i=1}^{k}\left[w_i^T C_{XY} w_i - w_i^T C_{X\widehat{Y}} w_i + \beta w_i^T C_X w_i\right], \tag{4.18}$$

where C_{XY} and $C_{X\widehat{Y}}$ denote the cross-covariance matrices between similar and dissimilar $\mathbf{X}$ and $\mathbf{Y}$ respectively, and $\beta > 0$ is a weighting parameter chosen by cross-validation. The optimal projection matrix W then consists of the top eigenvectors of the matrix $C_{XY} - C_{X\widehat{Y}} + \beta C_X$.

Under the second-order statistical model of Section 4.1, C_{XY} becomes C_X and $C_{X\widehat{Y}}$ is the zero matrix. As a result, the optimal projections of SSH are equivalent to those of PCA Hashing.

In the next section, we will compare the empirical performance of SNR-MH with that of PCAH, SSH, and other hashing algorithms.

4.3 Experimental Results and Discussion

4.3.1 Results on Synthetic Data

We first run simulations on synthetic datasets and compare SNR-MH and PCAH under the Gaussian model of Section 4.1.[2] We fix the feature dimension $d = 128$. The feature vector $\mathbf{X}$ consists of i.i.d. samples from $\mathcal{N}(\mathbf{0}, C_X)$. The covariance matrix $C_X = UD_XU^T$, where U is a random $d \times d$ orthogonal matrix and D_X is a $d \times d$ diagonal matrix with diagonal entries uniformly sampled from $(0.5, 1)$ and normalized so that their sum equals to $P = 128$, where P is the total signal power. The noise vector $\mathbf{Z}$ consists of i.i.d. samples from $\mathcal{N}(\mathbf{0}, C_Z)$ where $C_Z = VD_ZV^T$ where V is a random orthogonal matrix and $D_Z = \text{diag}\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$. Fixing the total noise power equal to P above, we consider three different scenarios depending on how $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ are designed:

1. **Uniform**: $d_{zi} = P/d, \ 1 \le i \le d$.
2. **Linear**: $d_{z_i} = a + (i-1)r, \ 1 \le i \le d$, where $a = 0.1$ and $\sum_{i=1}^{d} d_{zi} = P$.
3. **Exponential**: $d_{zi} = ar^{(i-1)}, \ 1 \le i \le d$, where $r = 1.05$ and $\sum_{i=1}^{d} d_{zi} = P$.

We generate 500,000 similar pairs for training, from which we estimate C_X and C_Z and solve (4.15) with these estimates. We generate another 500,000 similar and dissimilar pairs for testing. Simulation results are shown in Fig. 4.1. The left column shows SNR_i for each projection w_i learned by SNR maximization and PCA; the right column shows ROC curves for SNR-MH and PCAH at different code lengths. The rows corresponds to uniformly, linearly and exponentially generated $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ respectively.

Consider the left column first. As noted in Section 4.2, SNR-MH and PCAH coincide in the uniform scenario. In the linear (second row) and exponential (third row) scenarios where noise power is not evenly distributed,

[2]We only show results for PCAH in the synthetic experiments since PCAH and SSH are equivalent under the statistical model.

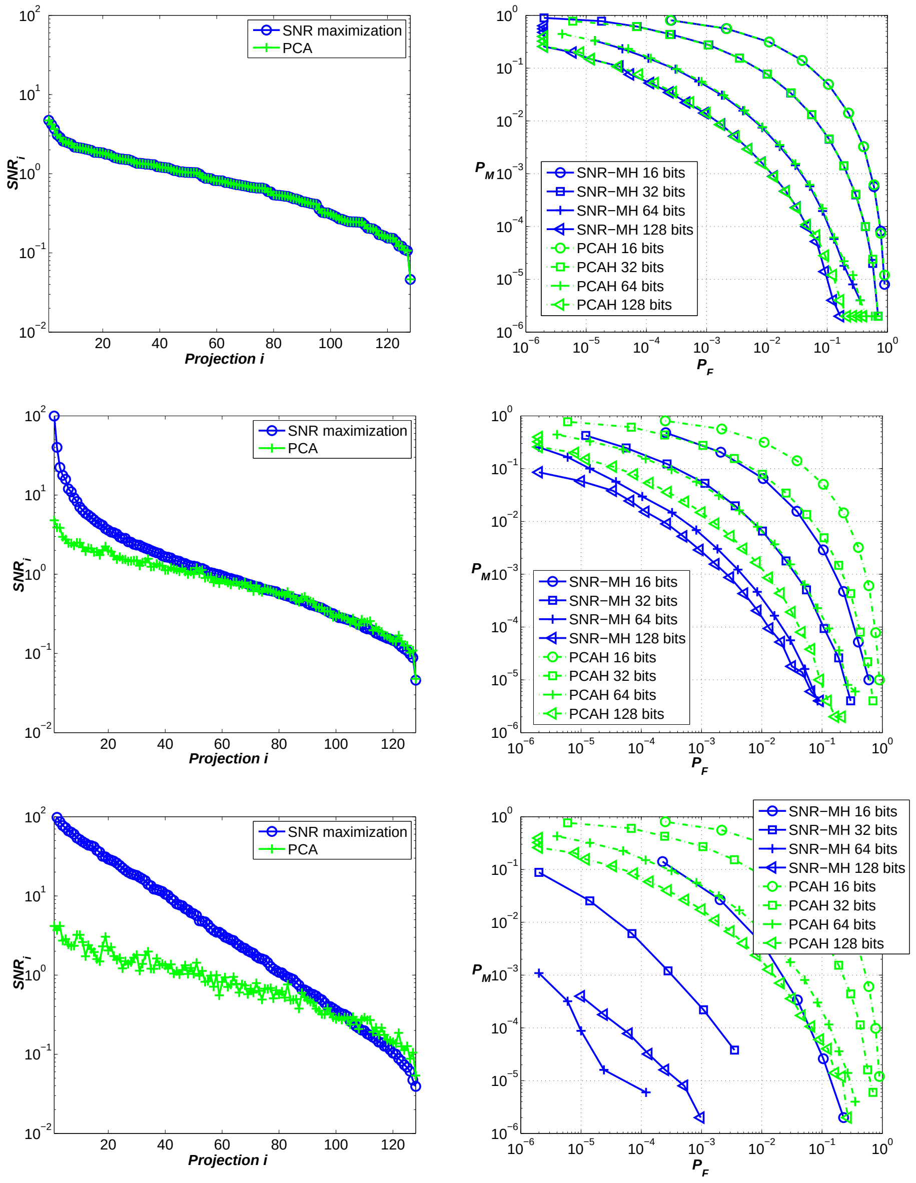


Figure 4.1: Experiments on the synthetic dataset. The left column shows SNR_i for each projection i learned by SNR maximization or PCA; the right column shows ROC curves for SNR-MH and PCAH at different code lengths. Rows corresponds to uniformly, linearly and exponentially generated $\{d_{z1}, d_{z2}, \ldots, d_{zd}\}$ respectively.

SNR_1 increases from 1.32 to 9.98 and 81.83 respectively in SNR maximization but remains largely unchanged in PCA. Additionally, more high-SNR projections are learned in the exponential case relative to the linear case because the number of small d_{zi}'s is larger. On the contrary, PCAH performs similarly across the three scenarios and SNR_i is generally not a monotone function of the PCA projections as PCAH seeks the variance-maximizing projections of the signal and ignores the noise structure.

To compare the hashing performance of the methods considered, we show ROC curves in the right column. We obtain $k+1$ points on the ROC curves, indexed by the integer threshold $\tau \in \{0, 1, \ldots, k\}$. The line segments between consecutive points are obtained by linear interpolation. They can be achieved by randomizing between the two deterministic tests corresponding to the endpoints of each such segment. Showing in the right column, SNR-MH and PCAH perform indistinguishably in the uniform scenario, whereas SNR-MH outperforms PCAH significantly in the linear and exponential scenarios, especially in the latter case where the gain is in orders of magnitude due to the high-SNR projections learned by SNR maximization.

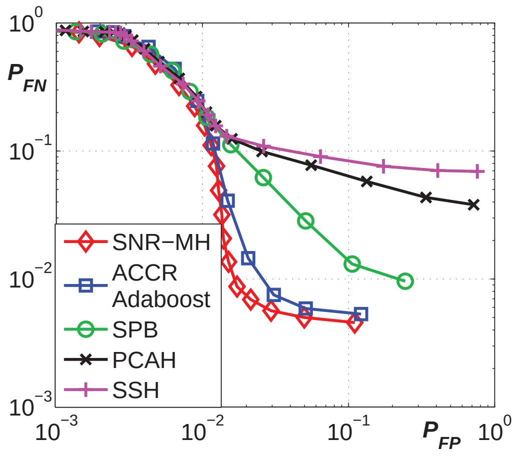
We notice that in the exponential scenario, 64-bit SNR-MH performs better than 128-bit SNR-MH. Though compact codes are more desirable in many applications, we expect performance to improve with longer codes. This behavior also arises in real datasets where error performance starts to deteriorate when more bits are added to the hashing codes. We will address this issue in more detail in Chapter 5 and propose strategies to keep improving performance with longer codes.

4.3.2 Results on Audio Content Identification

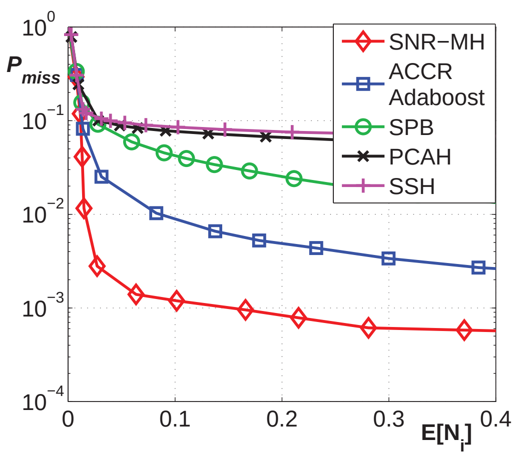
Next, we test our proposed SNR-MH on an audio content identification (ID) system. We follow the same experimental setup as in Section 3.6.2 for audio fingerprinting.

To compare performance, we estimate probability of false positive (P_{FP}) and probability of false negative (P_{FN}) for the single-output decoder, and expected number of incorrect items on the list ($\mathbb{E}(N_i)$) and probability of miss (P_{miss}) for the list decoder as defined in Section 3.2. Besides PCAH and SSH, we also compare with two boosting-based hashing algorithms, symmetric pairwise boosting (SPB) [28] and a regularized Adaboost (ACCR Adaboost) [57, 69], which have achieved excellent content ID performance on audio.

Fig. 4.2 shows the performance comparison on the audio content identification experiments. For both decoders, SNR-MH outperforms all other methods. For the list decoder, SNR-MH outperforms the next best by almost



(a) Single-Output Decoder.



(b) List Decoder.

Figure 4.2: Experiments on audio content identification. The query consists of 16 audio segments and 32 bits are extracted from each audio segment by each hashing algorithm.

an order of magnitude.

4.3.3 Results on Object Retrieval

We also evaluate SNR-MH on the University of Kentucky Object Recognition dataset [70]. There are 2,550 different objects in the dataset, each of which contains four images taken under different viewpoint, orientation, scale, or lighting conditions. See Fig. 4.3 for some example objects.

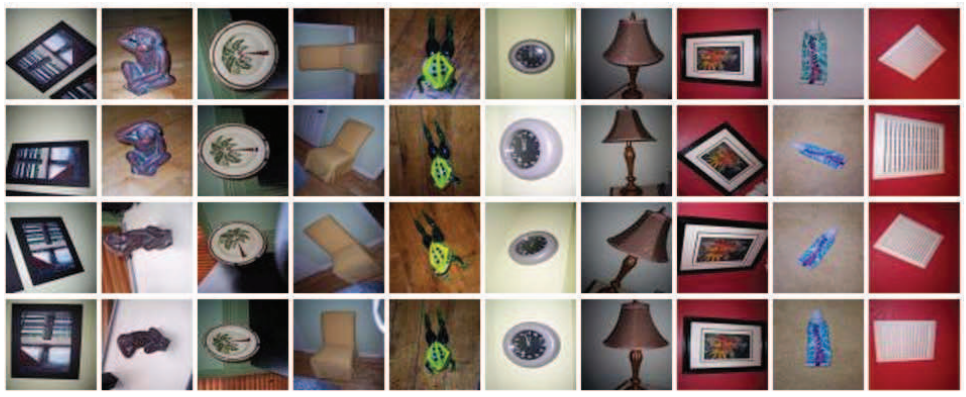
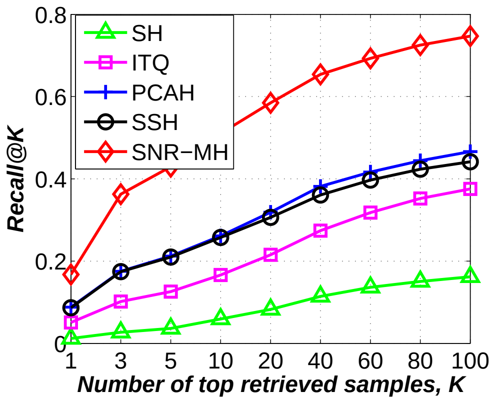


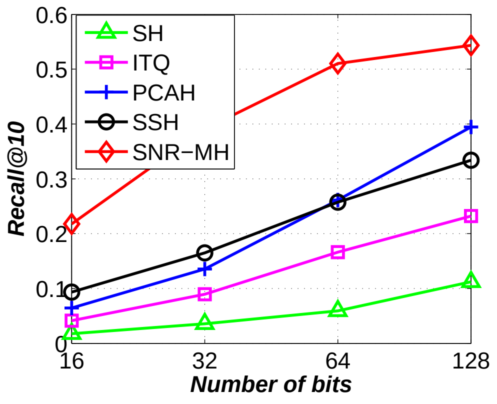
Figure 4.3: Examples from the University of Kentucky Object Recognition dataset.

Images in the dataset are 640 × 480 pixels. Each image is represented by a 512-dimensional bag-of-SIFT-features (BoSF) [71]. In BoSF, dense SIFT descriptors are first extracted from every 16×16 pixel patches over a grid with spacing of 8 pixels and assigned to 512 visual words learned by k-means clustering.

We randomly take one image from each object as query and the rest are used as database and training set. We compute the recall@K for each query, where K is the number of top retrieved samples based on the Hamming distance between the query and database samples, and we report the average recall@K over all queries.



(a)



(b)

Figure 4.4: Performance comparison of different hashing algorithms on object retrieval. Results in (a) are generated using 64 bits.

Fig 4.4 shows the performance comparison of SNR-MH and other hashing algorithms including spectral hashing (SH) [5] and iterative quantization (ITQ) [6]. Fig. 4.4a compares hashing algorithms for 64-bit codes, showing Recall@K vs K, while Fig. 4.4b shows Recall@10 as a function of code size. The figures show that SNR-MH outperforms all other algorithms by a large margin, and Recall@10 increases rapidly as code size increases.

CHAPTER 5

MULTI-BIT HASHING

In the previous sections, we have shown that SNR-MH learns compact binary codes that outperform other hashing algorithms on both synthetic datasets and real datasets. The ability to learn compact binary codes that preserve semantic similarity is extremely valuable in large-scale retrieval systems as compact binary codes are both search- and storage-efficient. However, in certain applications with a higher bit budget, finding high-SNR projections that are uncorrelated to the previous chosen ones may become challenging.

As observed by several authors [4, 6], the performance of training data dependent hashing algorithms[1] does not always improve with longer code length. One possible reason is that learning a large number of projections overfits the training dataset. For SNR-MH, another reason is the deteriorating effect of low SNR projections. We show this effect both theoretically and empirically, and propose a remedy which we call SNR multi-bit hashing (SNR-MBH). SNR-MBH is comprised of the following three components: (i) a simple iterative procedure that automatically determines the cutoff number of projections beyond which adding bits from the discarded low-SNR projections would hurt performance; (ii) a bit allocation strategy to allocate multiple bits to each high-SNR projection when the number of bits needed exceeds the number of projections; (iii) a multi-bit quantization scheme that assigns multiple bits to each projection. Experiments on a synthetic dataset and real datasets demonstrate the superior performance of SNR-MBH.

[1]Randomized algorithms such as locality sensitive hashing (LSH) [2] and shift-invariant kernels LSH (SKLSH) [4] are data-independent. Though data-independent algorithms enjoy the theoretical guarantee that the underlining metrics are increasingly well preserved as the code length increases, they require much longer codes to work well.

5.1 Deteriorating Effect of Low-SNR Projections

Some of our experiments have shown that when bits are extracted from low-SNR projections, performance deteriorates (see the last subfigure in Fig. 4.1 for an example). In theory, adding more bits can never hurt performance if the optimal decision rule is used. Under the Gaussian model in Section 4.1, the optimal decision rule is a likelihood ratio test (LRT). The loglikelihood ratio

$$\begin{aligned}\Lambda &= \sum_{i=1}^{k} \log_2 \frac{P_{F_iG_i}(f_i, g_i)}{P_{F_i}(f_i)P_{G_i}(g_i)} \\ &= 2k + \sum_{i=1}^{k} \left(1_{\{f_i \neq g_i\}} \log_2 p_i + 1_{\{f_i = g_i\}} \log_2(1 - p_i)\right) \\ &= C + \sum_{i=1}^{k} 1_{\{f_i \neq g_i\}} \log_2 \frac{p_i}{1 - p_i},\end{aligned}$$

where $C = 2k + \sum_{i=1}^{k} \log_2(1 - p_i)$ is a constant, is to be compared with a threshold. Hence the LRT can be expressed in terms of the weighted Hamming distance

$$\sum_{i=1}^{k} \left(\log_2 \frac{1 - p_i}{p_i}\right) 1_{\{f_i \neq g_i\}} \mathop{\gtrless}_{D}^{S} \tau, \tag{5.1}$$

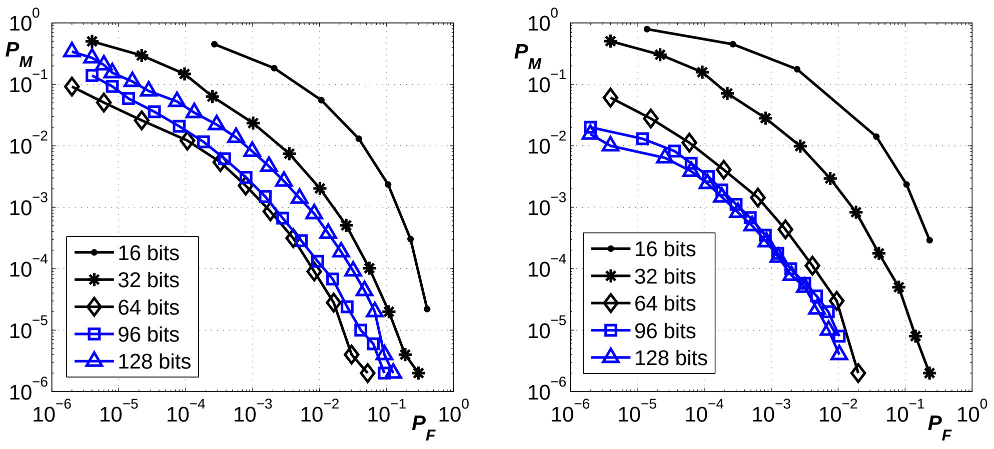
where τ is the threshold of the test and p_i is the probability that $F_i \neq G_i$ for similar fingerprints, as defined in (4.11). If $p_i = 1/2$, the weight $\log \frac{1-p_i}{p_i}$ is zero, rendering the i-th bit useless. For $p_i < 1/2$, a positive weight is assigned, and the smaller the p_i the larger the weight.[2]

On the other hand, the Hamming distance decision rule of (4.6) gives equal weight to each hash bit, which is a mismatched detector. The deteriorating effect of using bits from low-SNR projections can only be caused by this suboptimal decision rule.

To demonstrate the difference between the LRT of (5.1) and the suboptimal Hamming distance detector of (4.6), we run simulations on a synthetic dataset which is generated according to the 'exponential' scenario of Section 4.3.1, with the difference that the total noise power is five times of the total

[2]From (4.14), we know $p_i < \frac{1}{2}$. In practice, p_i is estimated from a training dataset. To avoid infinite weights, a small positive constant is added to the estimate.

signal power. As shown in Fig. 5.1, LRT and Hamming distance detector perform similarly up to 64 bits. Then, performance deteriorates from 64 bits to 96 bits and 128 bits for the Hamming distance detector, while performance keeps improving with longer codes for the LRT detector though the improvement is marginal.



(a) Hamming distance detector (4.6) (b) LRT (5.1)

Figure 5.1: ROC curves of SNR-MH on the synthetic dataset.

However, most hashing systems use Hamming distance because of its simplicity and search efficiency [1]. To enjoy these properties without sacrificing too much performance, we propose next a SNR multi-bit hashing (SNR-MBH) for which performance keeps improving with longer code length under the Hamming distance detector. In the following three subsections, we describe the three components of SNR-MBH, and in Section 5.5 we show SNR-MBH's superior performance on both the synthetic and real datasets. Note that SNR-MBH could also be used with LRT in a way similar to the model-based decision rule in [16], but we leave this for future work.

5.2 Cutoff Number of Projections

The first objective is to establish a simple procedure that determines whether including the $(k+1)$-th projection hurts performance. If so, we keep the first k projections only and discard the rest.

Denoting by T_k^D and T_k^S the Hamming distance between independent and related k-bit fingerprints respectively, then we express (4.7) and (4.8) in terms of the test threshold: $P_M^k(\tau) \triangleq \Pr\{T_k^S > \tau\}$ and $P_F^k(\tau) \triangleq \Pr\{T_k^D \leq \tau\}$. The goal is to determine the smallest $k \in \{1, 2, \dots, d\}$ such that for thresholds τ_k and τ_{k+1} corresponding to a fixed false alarm probability $P_F^k(\tau_k) = P_F^{k+1}(\tau_{k+1}) = \alpha$, the probability of miss increases from k to $k+1$: $P_M^k(\tau) < P_M^{k+1}(\tau_{k+1})$. This smallest k will be our cutoff number of projections and denoted by K_c.

Under the Gaussian model of Section 4.1, we have $T_k^D \sim \text{Bi}(k, \frac{1}{2})$ and $T_k^S \sim \text{PBD}(\{p_1, \dots, p_k\})$. However, working with the binomial distribution and PBD in determining K_c poses two challenges. First, computing P_M^k is infeasible for large k as the number of terms in PBD is combinatorial (4.12). Second, there generally exist no τ_k and τ_{k+1} achieving $P_F^k(\tau_k) = P_F^{k+1}(\tau_{k+1}) = \alpha$ due to the discrete nature of binomial distributions. To overcome these challenges, we use the fact that both T_k^D and T_k^S are sums of independent random variables. For large k, we use Gaussian approximations in the small- and moderate-deviations regime. The means of T_k^D and T_k^S are respectively $k/2$ and $\sum_{i=1}^k p_i$, and their variances are respectively $k/4$ and $\sum_{i=1}^k p_i(1-p_i)$. When the probabilities $P_F^k(\tau)$ and $P_M^k(\tau)$ are in the moderate-deviations regime (τ is a few standard deviations away from the mean), these two error probabilities can be reasonably well approximated using the Q-function.

Denote by

$$\widetilde{P_F^k}(\tau) \triangleq Q\left(\frac{k/2 - \tau}{\sqrt{k/4}}\right), \tag{5.2}$$

$$\widetilde{P_M^k}(\tau) \triangleq Q\left(\frac{\tau - \sum_{i=1}^k p_i}{\sqrt{\sum_{i=1}^k p_i(1-p_i)}}\right) \tag{5.3}$$

the Gaussian approximations of $P_F^k(\tau)$ and $P_M^k(\tau)$ respectively, and arrange p_i's in ascending order. We use the following procedure to determine K_c. For each $k = 1, 2, \dots, d-1$, obtain $\tau_k = k/2 - \sqrt{k/4}\,Q^{-1}(\alpha)$ and $\tau_{k+1} = (k+1)/2 - \sqrt{(k+1)/4}\,Q^{-1}(\alpha)$ from $\widetilde{P_F^k}(\tau_k) = \widetilde{P_F^{k+1}}(\tau_{k+1}) = \alpha$, and check whether the condition $\widetilde{P_M^{k+1}}(\tau_{k+1}) < \widetilde{P_M^k}(\tau_k)$ fails. The smallest k such that the condition fails will be the cutoff number of projections K_c. We keep the first K_c projections and discard the rest.

Throughout our experiments, we use $\alpha = 10^{-3}$ corresponding to a threshold τ that is about three standard deviations away from the mean $k/2$. Moreover, in calculating $\widetilde{P_M^k}(\tau)$, one needs to estimate $\{p_i\}$ from a training dataset. While based on approximations, this cutoff selection procedure worked well in our experiments.

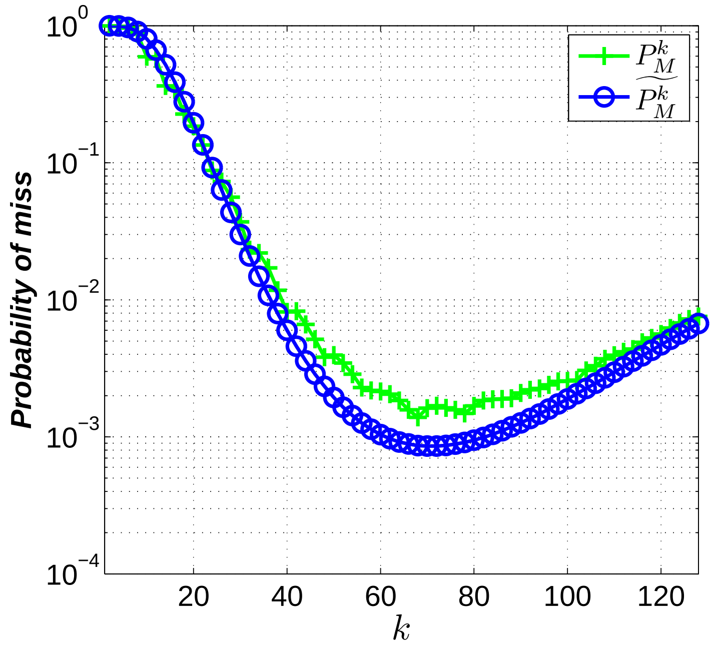


Figure 5.2: Comparison between probability of miss from simulation results on the synthetic dataset and their Gaussian approximations. Both P_F^k and $\widetilde{P_F^k}$ are fixed at 10^{-3}, where $P_F^k = 10^{-3}$ is obtained by a randomized decision rule.

To evaluate this procedure, we run simulations on the synthetic dataset of Section 5.1. We fix P_F^k and $\widetilde{P_F^k}$ at $\alpha = 10^{-3}$, where $P_F^k = 10^{-3}$ is achieved by the randomized version of the decision rule (4.6). As shown in Fig. 5.2, the Gaussian approximations are reasonably close to the simulated results. Moreover, P_M^k and $\widetilde{P_M^k}$ achieve their minima at similar values of k which is important because the minimum of $\widetilde{P_M^k}$ is our cutoff K_c. The cutoff number of projections returned by the procedure is $K_c = 72$, which indicates that using hash codes that are larger than 72 bits could hurt performance. Indeed, we see in Fig. 5.1a that 96-bit and 128-bit SNR-MH perform worse than 64-bit SNR-MH.

5.3 A Bit Allocation Strategy

To generate $N > K_c$ bits, one must extract more than one bit from some projections. Hence the first task is to decide how to allocate N bits across K_c projections. This type of problem often arises in information theory. In particular, for a $\mathcal{N}(0, \sigma^2)$ source, the rate distortion function $R(D) = \frac{1}{2} \log \frac{\sigma^2}{D}$ gives the minimum bit rate needed to describe the source with MSE not exceeding D [34]. This motivated us to develop a bit allocation strategy based on $\log \mathrm{SNR}_i$. Denote by

$$B_i = \left\lceil N \frac{\log(\mathrm{SNR}_i + 1)}{\sum_{i=1}^{K_c} \log(\mathrm{SNR}_i + 1)} \right\rceil \tag{5.4}$$

the number of bits projection w_i can accommodate, where $\lceil \cdot \rceil$ is the ceiling function and adding one to each SNR_i ensures each projection is allocated at least one bit. The total number of bits for the K_c projections is $B = \sum_{i=1}^{K_c} B_i \geq N$. If the inequality is strict, we need to prune $B - N$ bits from the total.

To do so, we first assign $N_i = B_i$ bits to each projection, and then prune bits from each projection until $\sum_{i=1}^{K_c} N_i = N$, as described in Table 5.1. The pruning follows two rules: (i) $N_i \geq N_j, \forall i < j$; (ii) whenever $N_j < B_j$, $N_i \leq N_j + 1, \forall i < j$. These two rules ensure that bits are spread as evenly as possible among the K_c projections while satisfying $N_i \leq B_i$.

Table 5.1: The bit allocation strategy.

```
Input: number of high-SNR projections $K$, bit budget $N$, and $B_i$ from (5.4) for $1 \leq i \leq K_c$.
Initialization: $N_i = B_i, \forall i$ and $M = \sum_{i=1}^{K_c} N_i - N$
while $M > 0$
        $i = \max\{j : N_j = \max\{N_k, k = 1, 2, \ldots, K_c\}\}$,
        $N_i - -$,
        $M = \sum_{i=1}^{K_c} N_i - N$.
Output: number of bits assigned to each projection $N_i, 1 \leq i \leq K_c$.
```

5.4 Multi-Bit Quantization

After bit allocation, we need to extract N_i bits from projection w_i. An N_i-bit scalar quantizer partitions the real line into 2^{N_i} bins separated by $2^{N_i} - 1$ thresholds. As N_i could be large, we propose a multi-bit quantization scheme where the number of thresholds grows linearly, rather than exponentially, with N_i. Motivated by the 2-bit quantization procedure of [28, 69], we define the i-th threshold, $i = 1, \ldots, N_i$, as the $i/(N_i + 1) \times 100\%$ quantile of the distribution of the transformed feature $w_i^T \mathbf{X}$. The N_i thresholds induce $N_i + 1$ bins, and we assign a length-N_i bit string to the t-th bin with $N_i + 1 - t$ zeros followed by $t - 1$ ones for $t = 1, \ldots, N_i + 1$. Besides the linear growth of the number of thresholds with N_i, another advantage is that the Hamming distance between the binary code for the t-th bin and the binary code for the $(t + s)$-th bin is exactly s, which makes the binary code distance preserving. The quantization scheme is illustrated in Fig. 5.3 with $N_i = 3$.

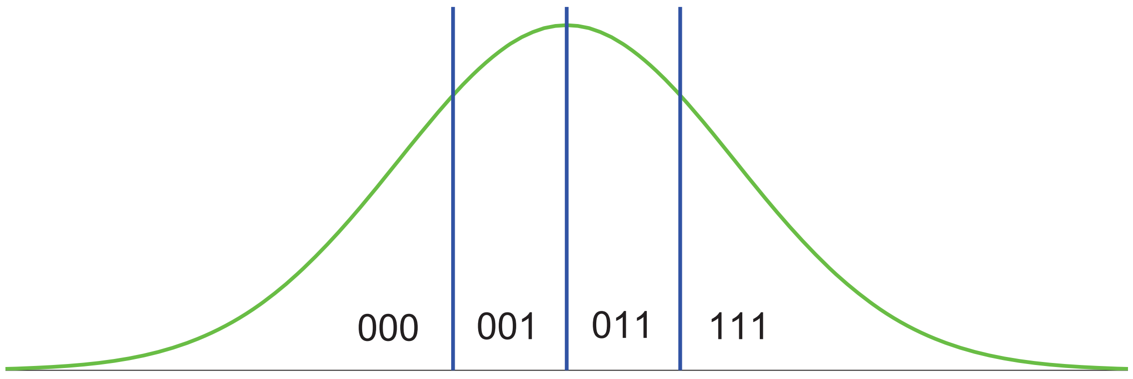


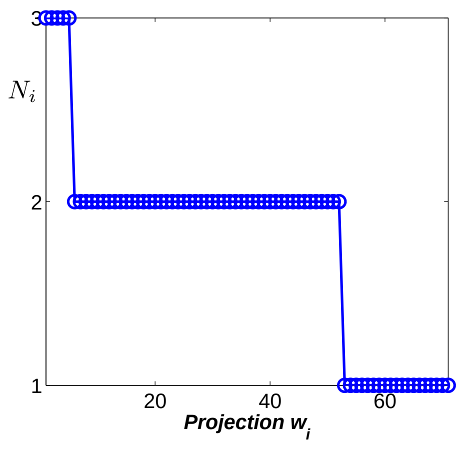
Figure 5.3: The quantization scheme. The three thresholds are the (25%, 50%, 75%) quantiles of the distribution.

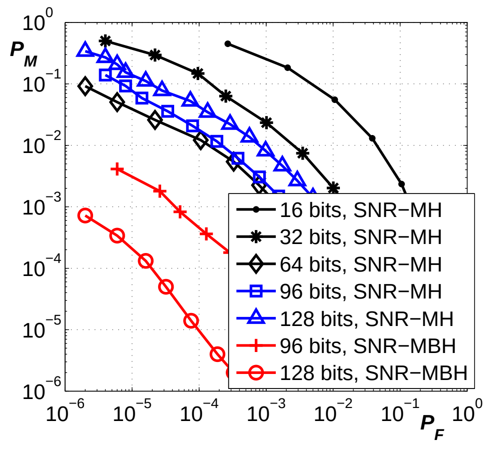
5.5 Experimental Results and Discussion

5.5.1 Results on Synthetic Data

We show results on the synthetic dataset of Section 5.1. We select $K_c = 72$ high-SNR projections as determined by the procedure described in Section 5.2. To generate 96 and 128 bits, we use the bit allocation strategy of Section

5.3 (shown in Fig. 5.4a is the bit allocation result for the 128-bit SNR-MBH) and the multi-bit quantization scheme of Section 5.4. As shown in Fig. 5.4b, 96-bit and 128-bit SNR-MBH using 72 projections outperform the corresponding SNR-MH using all 128 projections.





(a) Bit allocation for 128-bit SNR-MBH.

(b) ROC curves for SNR-MH and SNR-MBH.

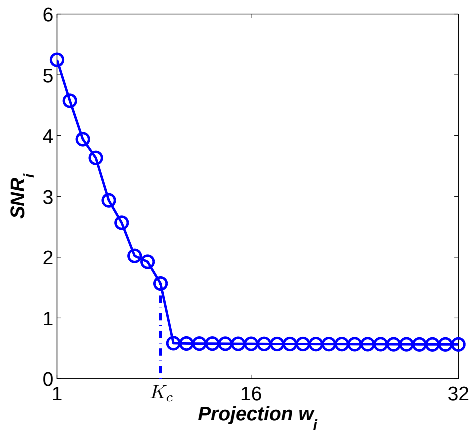
Figure 5.4: Experiments on the synthetic dataset generated the same way as Fig. 5.1. In (b), SNR-MBH uses only the first 72 projections to generate binary codes.

5.5.2 Results on MNIST Dataset

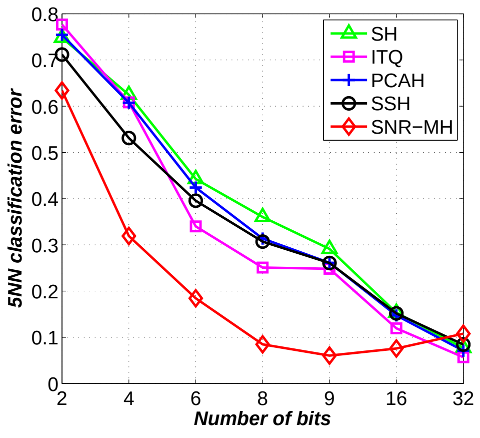
Next, we conduct experiments on the MNIST handwritten digit dataset[3] to demonstrate the power of SNR-MH to learn compact codes and SNR-MNH to learn longer codes. The MNIST dataset contains 60,000 training images and 10,000 testing images of ten handwritten digits. Each image is of size 28×28 pixels, from which we extract 512-dimensional GIST features [72].

As shown in Fig. 5.5a, the cutoff number of projections is very small $K_c = 9$, so we expect bits generated from the tenth projection onward start to hurt performance. Fig. 5.5b shows the 5 nearest neighbor (NN) classification performance, based on Hamming distance ranking, for different methods at different code lengths. SNR-MH performs impressively well with the first 9 bits. However, as we include more low SNR projections to generate hash bits, the performance of SNR-MH deteriorates.

[3]http://yann.lecun.com/exdb/mnist/

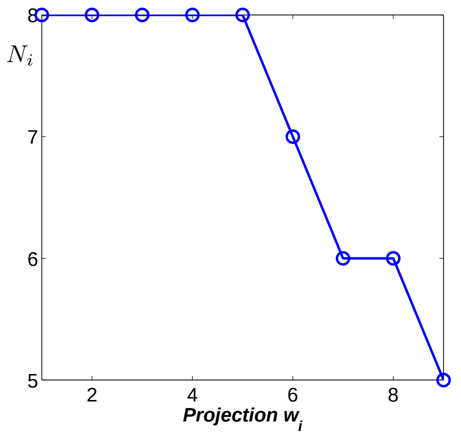


(a) SNR$_i$ for the first 32 projections where $K_c = 9$.

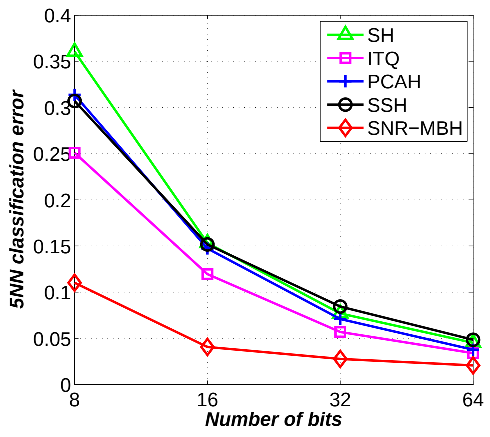


(b) 5NN classification error at different code lengths.

Figure 5.5: Experiments on the MNIST dataset.



(a) Bit allocation for 64-bit SNR-MBH.



(b) 5NN classification error.

Figure 5.6: Experiments on the MNIST dataset using 9 projections for SNR-MBH.

The bit allocation for 64-bit SNR-MBH with nine high-SNR projections is shown in Fig. 5.6a, and 5NN performance is shown in Fig. 5.6b. Now, the classification error keeps decreasing with longer codes and is lower than that for competing methods. SNR-MBH also exhibits superior performance in retrieval tasks, as shown in Fig. 5.7.

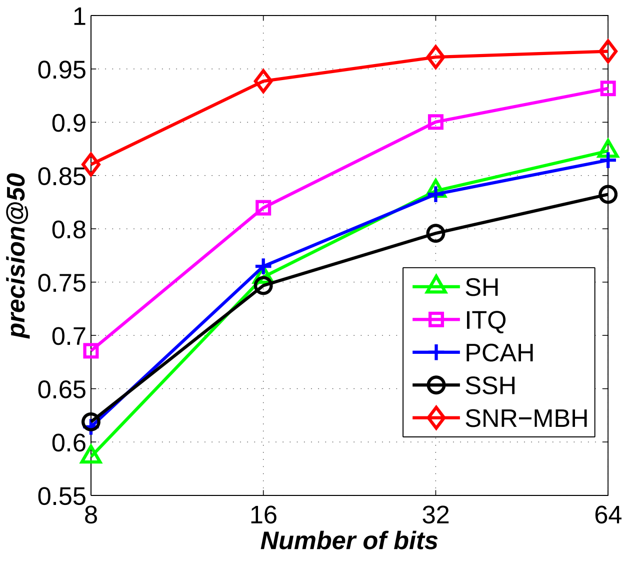


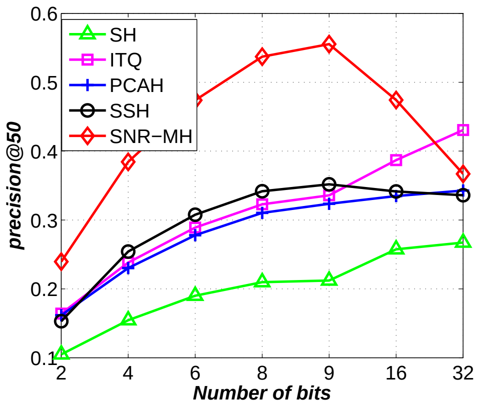
Figure 5.7: Retrieval performance on the MNIST dataset. SNR-MBH uses at most 9 projections.

5.5.3 Results on CIFAR-10 Dataset

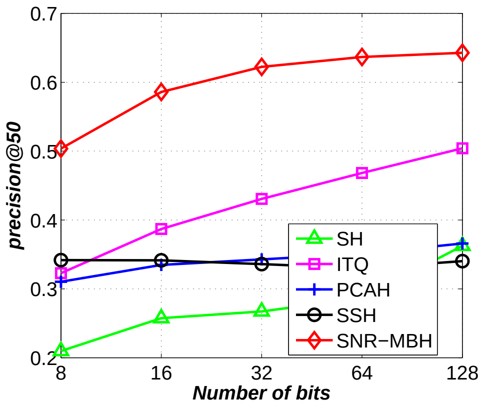
The CIFAR-10 dataset [73] consists of 50,000 training and 10,000 test color images of size 32×32 pixels. Images have been manually grouped into ten classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. From each image, we extract the state-of-the-art Convolutional Network-based image features using the feature extractor Overfeat [74], resulting in a 4,096-dimensional feature vector. Afterwards, we use PCA to reduce the feature dimension to 512, which retains 99.8% of the total signal variance.

Similar to the experiments on the MNIST dataset, $K_c = 9$ high-SNR projections are selected by the procedure in Section 5.2. As shown in Fig. 5.8a, the retrieval performance of SNR-MH drops drastically from 9-bit code to 16-bit code. However in Fig. 5.8b, we see a strong upward trajectory for SNR-MBH and it outperforms the next best by a large margin across different code lengths.

In Fig. 5.9, we also show three retrieval examples. In all three examples, SNR-MBH returns more true matches than ITQ. Moreover, SNR-MBH is able to retrieve matches that are not visually similar to the queries, such as the B-2 bombers to the first query of airplane.



(a) SNR-MH



(b) SNR-MBH using 9 projections.

Figure 5.8: Experiments on the CIFAR-10 dataset.

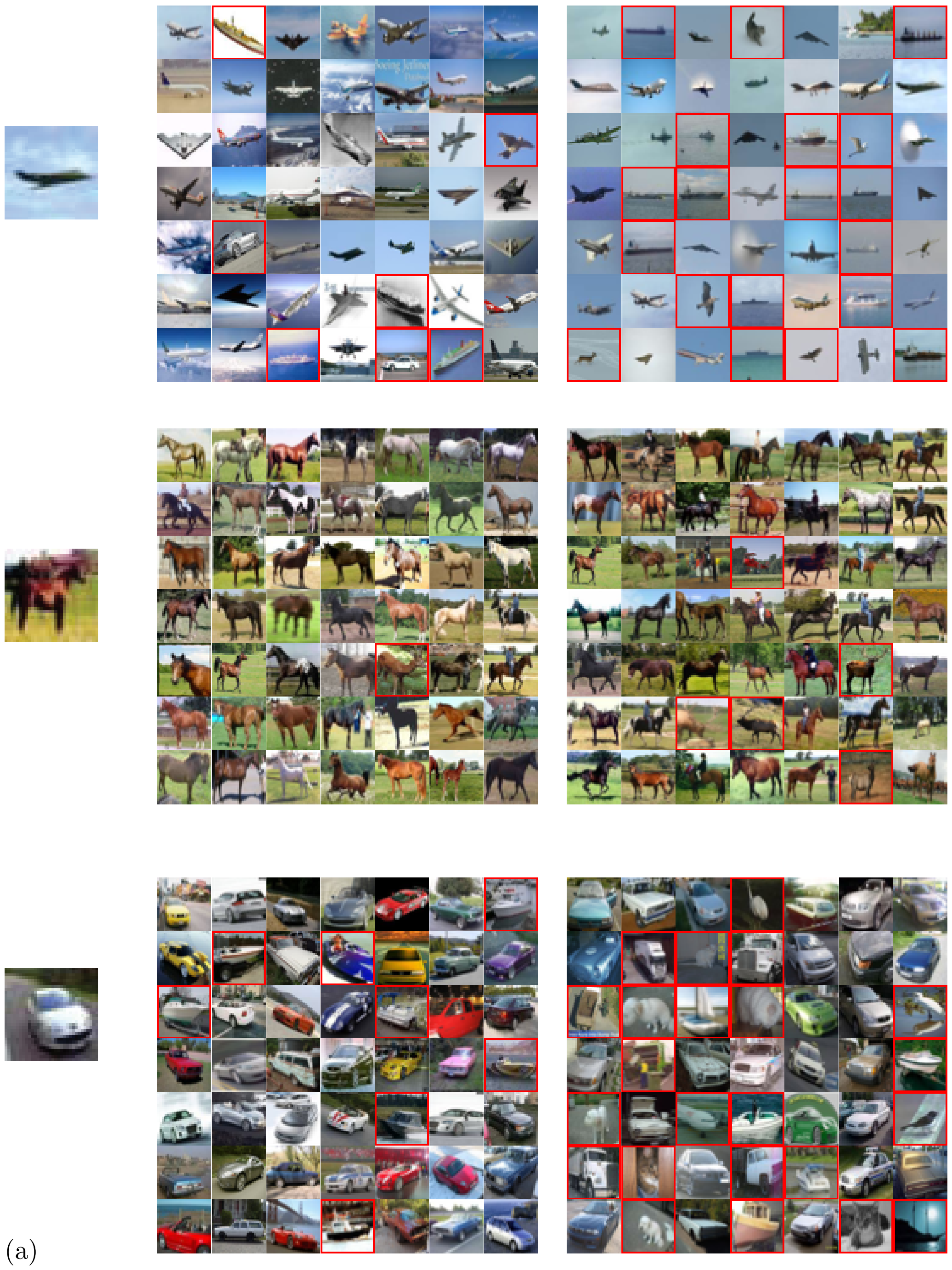


Figure 5.9: Retrieval examples from the CIFAR-10 dataset.

CHAPTER 6

MULTI-FEATURE HASHING

Despite the progress achieved in hashing-based similarity search, most existing methods only utilize one type of feature. However, different features extracted from the same underlining signal can be complementary to each other and boost system performance. For instance, combining raw image pixels with patch-based features gives better face recognition performance [75]. In near-duplicate video retrieval, combining features such as color histogram and local binary pattern yields a significant performance improvement [76]. Recently, fusion of RGB and depth features has achieved state-of-the-art results in many vision tasks, such as object recognition [46], indoor scene segmentation [44], and video content identification [77].

In the literature, not much work has been reported regarding multi-feature hashing except [78, 76, 65, 79]. In [78, 65] the final hash function is the convex combination of individual hash functions on different features, where hash functions in [78] are linear, whereas [65] learns nonlinear hash function on each feature using the kernel trick. In [76], features are concatenated and binary codes are generated using learned hyperplanes to simultaneously preserve similarities in each individual feature space. In contrast to the other three, [79] selects hash bits from a pool of hash bits generated by different hashing methods on different features. The more recent works on multi-feature kernel hashing (MFKH) [65] and hash bit selection (HBS) [79] have shown superior performance over previous art [78, 76].

Unlike multi-feature hashing, multimodal hashing for cross-modality similarity search between text and images has been well studied [80, 81, 82]. Here, queries are in one modality (tags or images) while database items are in the other modality (images or tags). In contrast, all features are used for both database indexing and query search in multi-feature hashing.

In this chapter, we propose two multi-feature hashing methods based on signal-to-noise ratio (SNR) maximization, which has been shown to be equivalent to the minimization of hashing error probability under a Gaussian model. The first method concatenates different features as one and jointly learns uncorrelated hash functions that maximize SNR. We call this method SNR joint hashing (SNR-JH). The second method separately learns hash functions on each feature based on SNR maximization and the overall hash functions are selected according the SNR associated with each hash function. We call this selection procedure SNR selection hashing (SNR-SH). Both SNR-JH and SNR-SH outperform the state-of-the-art MFKH and HBS significantly on several benchmark datasets.

6.1 Background and Related Work

In multi-feature hashing, the basic task is to learn a mapping $h(\mathbf{x}) = \{h_1(\mathbf{x}), \ldots, h_k(\mathbf{x})\} \in \{\pm 1\}^K$ that projects an input $\mathbf{x} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}\}$ consisting of M features, each having dimensionality d_m, onto K-dimensional binary codes, while preserving some notion of similarity. A simple way to learn this mapping is to concatenate different features, treat them as one, and apply the previous single-feature hashing methods. However, without considering the different statistical properties from different features, most existing single-feature hashing methods would perform poorly (often worse than the best single feature). In the following two subsections, we briefly summarize two state-of-the-art multi-feature hashing methods, multi-feature kernel hashing and hash bit selection.

6.1.1 Multi-Feature Kernel Hashing

Inspired by multiple kernel learning, Multi-Feature Kernel Hashing (MFKH) [65] formulates the hashing problem as a similarity preserving hashing with linearly combined multiple kernels. In particular, each input $\mathbf{x}^{(m)}$ is implicitly embedded in the high dimensional (possibly infinite dimensional) feature space by an embedding function $\phi_m : \mathbb{R}^{d_m} \to \mathcal{F}^{(m)}$, and the overall embedding $\phi(\mathbf{x}) = [\mu_1^{\frac{1}{2}} \phi_1^T(\mathbf{x}^{(1)}), \ldots, \mu_M^{\frac{1}{2}} \phi_M^T(\mathbf{x}^{(M)})]^T$ is a weighted concatenation of $\{\phi_m(\mathbf{x}^{(m)})\}$. Therefore, the kernel function $K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \sum_{m=1}^{M} \mu_m K_{ij}^{(m)}$ a linear combination of the kernels on different features. With the embedding function $\phi(\cdot)$, the k-th hash function is defined as

$$h_k(\mathbf{x}) = \operatorname{sgn}\left(v_k^T \phi(\mathbf{x}) + b_k\right), \tag{6.1}$$

where v_k^T is a projection vector in the high dimensional feature space and b_k is the bias.

Similarly to spectral hashing, the hash codes for N training data are learned to preserve a similarity matrix $S \in \mathbb{R}^{N\times N}$ while satisfying the balance and uncorrelated constraints. Formally, MFKH formulates the multi-feature hashing problem as follows:

$$\begin{aligned}
\min_{W,\mathbf{b},\boldsymbol{\mu}} \quad & \frac{1}{2}\sum_{i,j=1}^{N} S_{ij}||\mathbf{y}_i - \mathbf{y}_j||^2 + \lambda||V||_F^2 \\
s.t. \quad & \mathbf{y}_i \in \{\pm 1\}^K \\
& \sum_{i=1}^{N}\mathbf{y}_i = \mathbf{0}, \quad \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i\mathbf{y}_i^T = I \\
& \mathbf{1}^T\boldsymbol{\mu} = 1, \quad \boldsymbol{\mu} \succeq \mathbf{0},
\end{aligned} \tag{6.2}$$

where $\mathbf{y}_i = h(\mathbf{x}_i)$, λ is a regularization parameter, $v_k = \sum_{l=1}^{L} W_{lk}\phi(\mathbf{p}_l)$ is represented as a combination of L landmarks $\mathbf{p}_l$ embedded in the feature space, and $W \in \mathbb{R}^{L\times K}$ is a weight matrix. By using the kernel trick, one only needs to evaluate kernel functions instead of working with the high dimensional feature embedding $\phi(\cdot)$.

The motivation for such formulation is to preserve the given similarity S between examples in the Hamming space. However, due to the nondifferentiable and nonconvex sgn function, the above optimization is difficult to solve. Similarly to spectral hashing, the sgn function is replaced with the identity function, after which the minimization problem can be efficiently solved using an alternating minimization procedure. Though approximating the sgn function with the identity function makes the optimization tractable, it obviously introduces large approximation error when the magnitude is large.

6.1.2 Hash Bit Selection

Hash Bit Selection (HBS) [79] is a unified framework for various selection problems in hashing. Here, we limit our attention to the scenario of hashing with multiple features. In contrast to MFKH, in which bits are generated from the combination of all features, each bit in HBS is derived from only one type of feature. In particular, HBS selects K hash bits (corresponding to K hash functions) from a pool of candidate bits generated by a given hashing method using different features.

In HBS, the selection criteria are similarity preservation and independence. Similar to spectral hashing and MFKH, similarity preservation means the binary codes should preserve the original similarity measure S between training data points in the Hamming space, and HBS uses a loss function based on spectral embedding loss [5]. In [5, 12, 65, 79], the independence of hash functions is considered a desirable property for generating compact binary codes, and HBS measures the independence using pairwise mutual information between hash bits. Combining these two criteria, HBS formulates the bit selection problem as quadratic programming. By relaxing the discrete constraint, it can be solved by replicator dynamics [79].

6.2 SNR Joint Hashing

Similarly to many other hashing methods [9, 12, 10, 7, 55], the training dataset for SNR-MH requires weakly supervised information in the form of N similar feature pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^d$. Without loss of generality, we assume that feature vectors are zero-centered, i.e., $\sum_{i=1}^N \mathbf{x}_i = \mathbf{0}$ and $\sum_{i=1}^N \mathbf{y}_i = \mathbf{0}$. A pair $(\mathbf{x}_i, \mathbf{y}_i)$ is said to be *similar* if $\mathbf{y}_i$ is a distorted version of $\mathbf{x}_i$, or $\mathbf{x}_i$ and $\mathbf{y}_i$ share the same class label. We assume $\mathbf{y}_i$ is the sum of $\mathbf{x}_i$ and independent noise $\mathbf{z}_i$ as governed by (4.3). For *dissimilar* pairs of $\mathbf{x}_i$ and $\mathbf{y}_i$, they are assumed to be independent. Operating under this assumption, SNR-MH does not require dissimilar pairs for training unlike methods in [9, 12, 10, 7, 55].

We propose the first SNR-based multi-feature hashing method, SNR joint hashing (SNR-JH), to learn hash functions from the combination of all features. In a multiple feature setting, feature vectors $\mathbf{x} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}\}$ and $\mathbf{y} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(M)}\}$, both zero-centered, consist of M different features. Similarly to SNR-MH, the goal of SNR-JH is to learn K projection vectors from M different features such that the projected feature vector is uncorrelated and SNR at each projection is maximized.

Following composite hashing in [78] and MFKH, hash functions h_k in SNR-JH is parameterized by projection vectors $\{w_k^{(m)}\}_{m=1}^M$ on each individual feature m and non-negative weights $\{\mu_m\}_{m=1}^M$ that linearly combine different features:

$$h_k(\mathbf{x}) = \text{sgn}\left(\sum_{m=1}^{M} \mu_m w_k^{(m)^T} \mathbf{x}^{(m)}\right), \tag{6.3}$$

where $\sum_{m=1}^M \mu_m = 1$ and $w_k^{(m)} \in \mathbb{R}^{d_m}$.

In both composite hashing and MFKH, learning of $\{w_k^{(m)}\}_{m=1}^M$ and $\{\mu_m\}_{m=1}^M$ is done via alternating optimization as the objective functions are nonconvex with respective to $\{w_k^{(m)}\}_{m=1}^M$ and $\{\mu_m\}_{m=1}^M$ jointly. However in SNR-JH, we show that the weights $\{\mu_m\}_{m=1}^M$ are redundant and can be incorporated into $\{w_k^{(m)}\}_{m=1}^M$.

Denote by

$$w_k = [\mu_1 w_k^{(1)^T}, \mu_2 w_k^{(2)^T}, \ldots, \mu_M w_k^{(M)^T}]^T \tag{6.4}$$

the k-th projection direction. Then we can rewrite (6.3) as $h_k(\mathbf{x}) = \text{sgn}\left(w_k^T \mathbf{x}\right)$. Also denote by

$$C_X = \begin{pmatrix} C_X^{(11)} & \ldots & C_X^{(1M)} \\ \vdots & \ddots & \vdots \\ C_X^{(M1)} & \ldots & C_X^{(MM)} \end{pmatrix} \tag{6.5}$$

and

$$C_Z = \begin{pmatrix} C_Z^{(11)} & \ldots & C_Z^{(1M)} \\ \vdots & \ddots & \vdots \\ C_Z^{(M1)} & \ldots & C_Z^{(MM)} \end{pmatrix} \tag{6.6}$$

the full covariance matrices among M feature vectors and noise vectors respectively, where $C_X^{(mn)} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{(m)} \mathbf{x}_i^{(n)^T}$ and $C_Z^{(mn)} = \frac{1}{N}\sum_{i=1}^N \left(\mathbf{y}_i^{(m)} - \mathbf{x}_i^{(m)}\right) \left(\mathbf{y}_i^{(n)} - \mathbf{x}_i^{(n)}\right)^T$, $m, n = 1, 2, \ldots, M$.

Therefore, $w_k, k = 1, 2, \ldots, K$ can be learned by solving (4.15) with C_X and C_Z given by (6.5) and (6.6) respectively. Clearly, SNR-JH does not need to learn the weights $\{\mu_m\}_{m=1}^M$ explicitly. The optimal linear combination of different features is automatically determined in SNR-JH.

To illustrate how SNR-JH allocates different weights to different features,

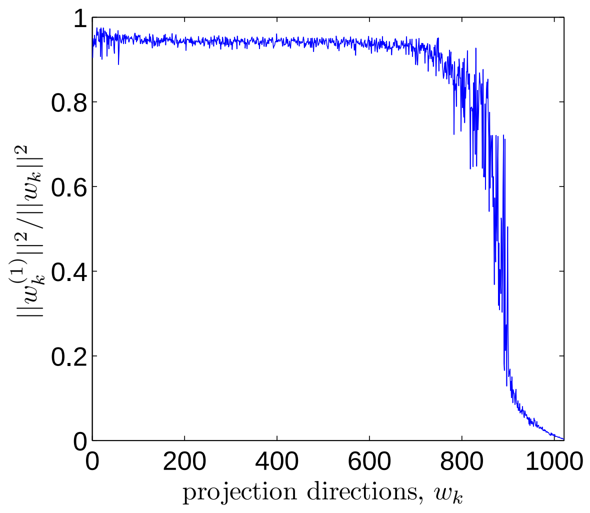


Figure 6.1: The ratio between energy allocated to feature 1 and the total energy of projection w_k. $\{w_k\}$'s are learned from the UK object recognition dataset.

Fig. 6.1 shows $||w_k^{(1)}||^2/||w_k||^2$, i.e., the ratio between energy allocated to feature 1 and the total energy of projection w_k, where $\{w_k\}$'s are learned from the UK object recognition dataset. As shown in Fig. 6.3, using feature 1 (the BoSF feature) alone yields much better performance than feature 2 (the GIST feature) for up to 128 bits, which explains why SNR-JH allocates more than 90% of the energy to feature 1 for the first few hundreds of projections. After around $k = 700$, the ratio falls rapidly, which indicates that low-SNR projections are comprised mostly of feature 2.

6.3 SNR Selection Hashing

The second SNR-based multi-feature hashing is a selection procedure, termed SNR selection hashing (SNR-SH). In SNR-SH, K projection directions are learned separately from each feature by solving (4.15). Unlike HBS where the selection criteria are similarity preservation and independence, SNR-SH uses the SNR in the projection direction as the only selection criterion.

On the m-th feature, we extract K projections $\{w_k^{(m)}\}_{k=1}^K$ as the top K eigenvectors of the generalized eigenproblem

$$C_X^{(m)} w^{(m)} = \gamma^{(m)} C_Z^{(m)} w^{(m)}, \tag{6.7}$$

where $\gamma_k^{(m)}$ is the SNR in the direction $w_k^{(m)}$, in descending order $\gamma_1^{(m)} \geq \gamma_2^{(m)} \geq \ldots \geq \gamma_K^{(m)}$. Among the candidate pool of projection directions $w_k^{(m)}, k = 1, \ldots, K, m = 1, \ldots, M$, SNR-SH selects K projections corresponding to the K largest $\gamma_k^{(m)}, k = 1, \ldots, K, m = 1, \ldots, M$.

It follows directly from (4.15) that $\{w_k^{(m)}\}_{i=1}^k$ learned on the m-th feature give us uncorrelated projection directions , i.e., $w_i^{(m)^T} C_X^{(m)} w_j^{(m)} = 0$ for $i \neq j$. However, $w_i^{(m)^T} C_X^{(mn)} w_j^{(n)}$ is not zero for $m \neq n$ in general unless features m and n are uncorrelated, i.e., $C_X^{(mn)}$ is the zero matrix. Therefore, bits generated from different features are in general correlated. In contrast, projection directions learned by SNR-JH are uncorrelated. Uncorrelatedness of hash bits is often considered desirable for generating compact binary hash codes [5, 65, 79].

To see the connection between SNR-SH and SNR-JH, let $\widetilde{C_X}$ and $\widetilde{C_Z}$ be the covariances obtained by forcing all off-diagonal sub-matrices of C_X and C_Z to zero. Then SNR-SH is equivalent to finding the top K eigenvectors of the following generalized eigenproblem

$$\widetilde{C_X} w = \gamma \widetilde{C_Z} w. \tag{6.8}$$

As SNR-JH jointly considers all the correlation structures among different features, more high SNR projections can be obtained than SNR-SH. On the other hand, SNR-JH needs to estimate considerably more parameters (all $M \times M$ sub-matrices $C_X^{(mn)}$ and $C_Z^{(mn)}$), which makes SNR-JH computationally less attractive than SNR-SH.

Learning projections in SNR-JH and SNR-SH is carried out by solving generalized eigenproblems, which can be done in less than a minute from a training dataset of 100,000 feature vectors of 1,000 dimension on a standard office desktop. Moreover, unlike MFKH and HBS where one needs to tune multiple parameters, both SNR-JH and SNR-SH are parameter-free, which makes training much easier.

6.4 Experimental Results and Discussion

6.4.1 Protocols and Baseline Methods

We compare SNR-JH and SNR-SH to two state-of-the-art multi-feature hashing methods MFKH and HBS using code provided by the authors. We also compare with other well-known single-feature hashing methods, iterative quantization (ITQ) and semi-supervised hashing (SSH), where different features are concatenated as one and treated as a single feature. ITQ and SSH are also served as the base methods for HBS, where a candidate pool of hash function are generated from each feature by ITQ and SSH, and later chosen by HBS. Whenever there are parameters in these baseline methods, we try a few candidate choices and report the best one. For MFKH, we consider three different kernels: linear, Gaussian and Chi-Square.

For evaluation, we compute the Recall@K or Precision@K for each query, where K is the number of top retrieved images based on the Hamming distance between the query and database images, and we report the average over all queries. We also compute the mean average precision (mAP), or the area under the precision-recall curve, for different code lengths. Besides retrieval performance, we also demonstrate the power of multi-feature hashing in terms of K-nearest neighbor (K-NN) classification performance on the MNIST digit dataset.

6.4.2 Datasets and Features

We perform experiments on five benchmark datasets: (1) the University of Kentucky (UK) object recognition dataset [70] contains 2,550 different objects, each of which contains four images of size 640×480 taken under different viewpoint, orientation, scale or lighting conditions; (2) the MNIST digit dataset [83] contains 70,000 images of ten handwritten digits; (3) CIFAR-10 [73] consists of 60,000 color images of size 32×32 pixels which have been manually grouped into ten classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck; (4) Scene-15 [84] contains fifteen natural scene categories, each of which has 200 to 400 images; (5) Caltech-101 [85] contains 101 categories, each of which has about 40 to 800 images.

For the UK object recognition dataset, we randomly select one image from each object (2,550 in total) as queries, and the rest are used for training and as database against which the queries are performed. For the other three datasets, we randomly select 1,000 images as queries, and the rest are used as training and database.

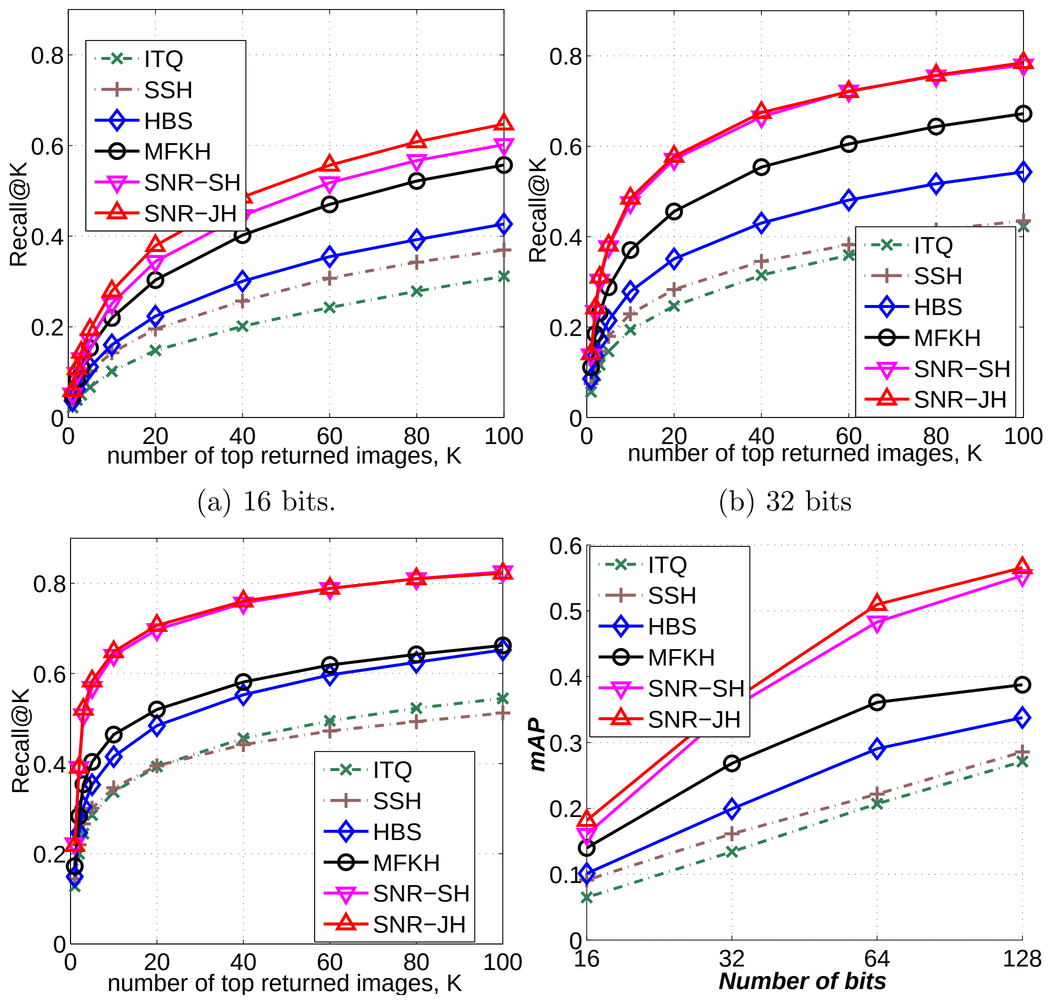
Images in the datasets are represented by the following three features: 512-D GIST feature [72], 512-D bag-of-SIFT-features (BoSF) [71], and 512-D convolutional neural networks (CNN) features. GIST features are computed at 8 orientations and 4 different scales, resulting in 512-dimensional feature vectors. In BoSF, SIFT descriptors are first extracted from every 16×16 pixel patches over a grid with spacing of 8 pixels and assigned to 512 visual words learned by k-means clustering. CNN features are extracted using Overfeat [74], resulting in 4,096-D feature vectors. Afterwards, we use principal component analysis (PCA) to reduce the feature dimension to 512, which retains about 99% of the total signal variance in our experiments. We limit our experiments to two features on each dataset. For all the multi-feature methods considered, it is trivial to extend to more than two features. The dataset-feature combination is shown in Table 6.1.

Table 6.1: The dataset-feature combination.

	UK	MNIST	CIFAR-10	Scene-15	Caltech-101
Feature 1	BoSF	GIST	GIST	GIST	CNN
Feature 2	GIST	CNN	CNN	BoSF	BoSF

6.4.3 Results and Discussions

Fig. 6.2a to Fig. 6.2c compare all the multi-feature hashing methods at different code lengths on the UK object recognition dataset, showing Recall@K vs K. First notice that both ITQ and SSH perform poorly on this multi-feature setting, but combining with the selection procedure HBS, we see a large performance improvement. Our proposed SNR-JH and SNR-SH both outperform the next best MFKH method by a large margin. Fig. 6.2d shows mAP as a function of code size. We see a strong upward trajectory for all methods. Again, our proposed SNR-JH and SNR-SH methods consistently outperform the other methods, and the performance gap to the next best method widens as more bits are used.



(a) 16 bits. (b) 32 bits

(c) 128 bits (d) mAP

Figure 6.2: Comparison with state-of-the-art methods on the UK object recognition dataset.

We also compare our multi-feature SNR-JH and SNR-SH methods with the single-feature counterpart SNR-MH on the UK object recognition dataset. As shown in Fig. 6.3, performance with multiple features is much better than with single feature, which indicates that our multiple feature methods help improve retrieval performance by exploiting the complementary information between features. As illustrated in Fig. 6.1, SNR-JH allocates a much larger weight on the BoSF feature, which exhibits superior performance over the GIST feature on this dataset. Moreover, SNR-SH selects 47 projections from BoSF and 17 projections from GIST for the 64-bit codes.

Displaying the retrieval results on the other four datasets, Fig. 6.4 to

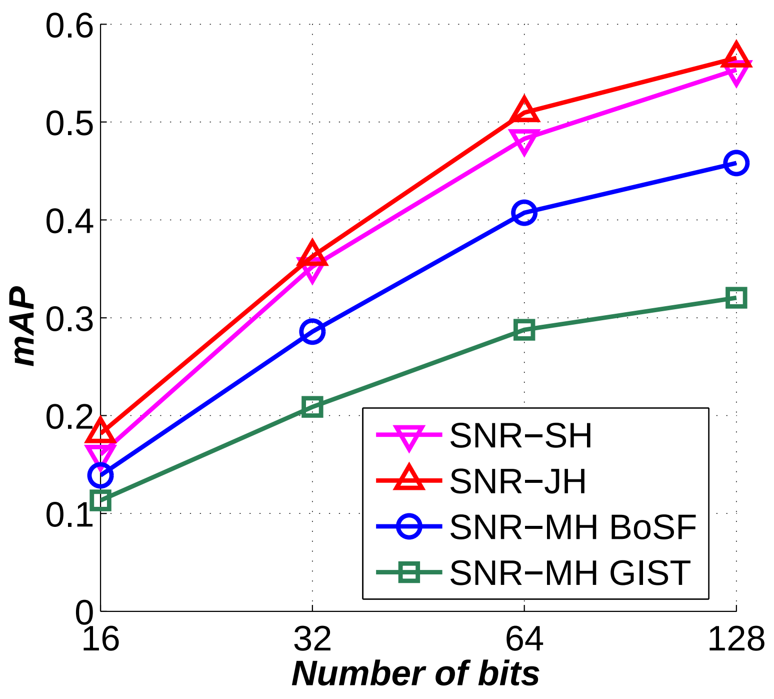
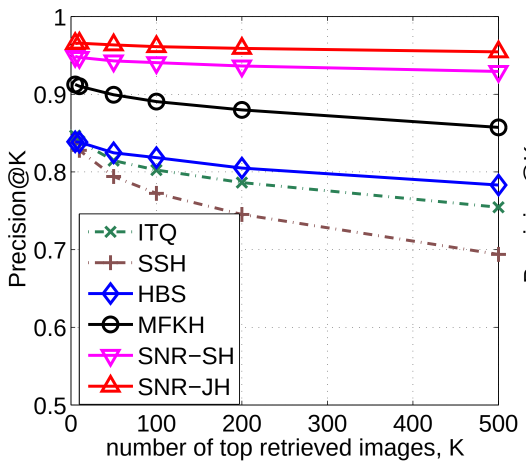


Figure 6.3: Comparison of multi-feature SNR-JH and SNR-SH with their single-feature counterparts SNR-MH on the UK object recognition dataset.

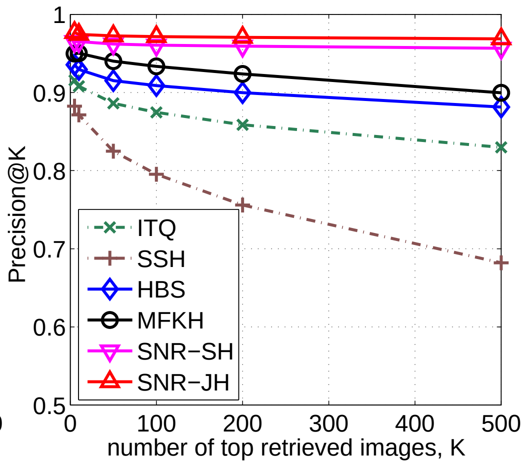
Fig. 6.7 show SNR-JH and SNR-SH's superior Recall@K or Precision@K performances over all other methods at all code lengths. Moreover, longer code length leads to larger performance gap between our proposed methods and the next best one. As shown in the mAP plots, SNR-JH and SNR-SH scale well with code length, which does not hold for MFKH and HBS.

For datasets such as CIFAR-10, images exhibit large intra-class variation, which poses a significant challenge for retrieval. However, good features, such as CNN, BoSF, and GIST, often make different classes linearly separable, which allows SNR-JH and SNR-SH's linear model of (4.3) to work well.

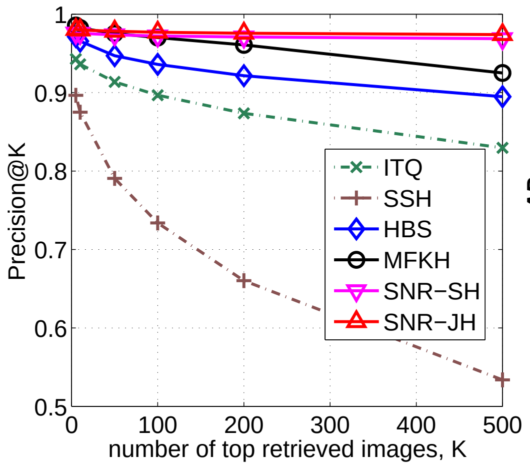
In addition to the retrieval performance, we also demonstrate the classification performance on the MNIST digit dataset. In this setting, we use the default split of the dataset, i.e., 60,000 images for training and 10,000 images for testing. The 30-NN classification results, based on Hamming distance ranking, are shown in Fig. 6.8. Using both GIST and CNN features, SNR-JH and SNR-SH improve upon their single-feature counterpart. Moreover, SNR-JH outperforms SNR-SH by a large margin across different code lengths. This may due to the following two reasons: (i) as SNR-JH jointly considers all the correlation structures among different features, more high



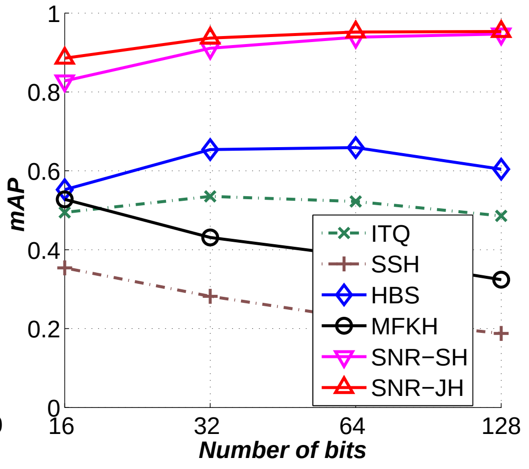
(a) 16 bits.



(b) 32 bits



(c) 128 bits

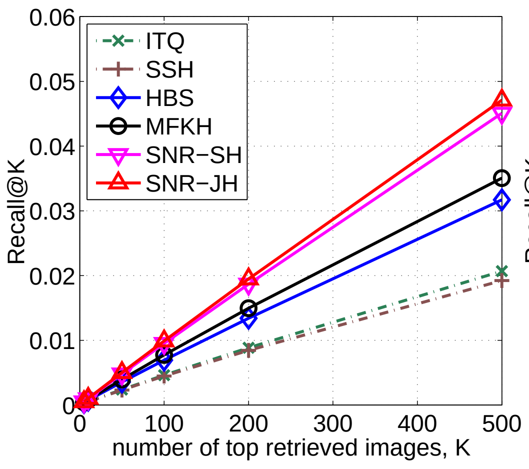


(d) mAP

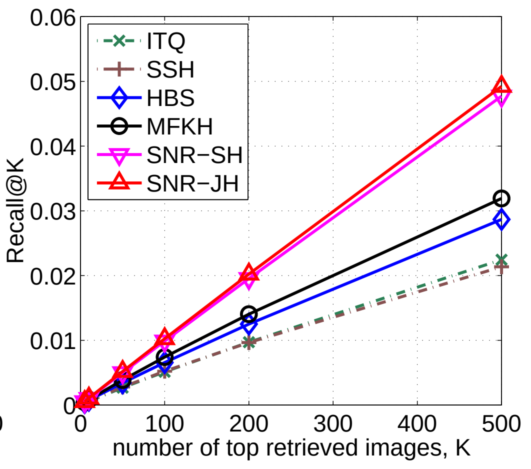
Figure 6.4: Comparison with state-of-the-art methods on the MNIST digit dataset.

SNR projections can be obtained than SNR-SH; (ii) the SNR-JH is able to learn uncorrelated projection directions, while projections from different features are correlated in SNR-SH. Strikingly, 128-bit SNR-JH achieves a 30-NN classification error rate of only 1.13%, which outperforms many sophisticated discriminative classifiers [7].

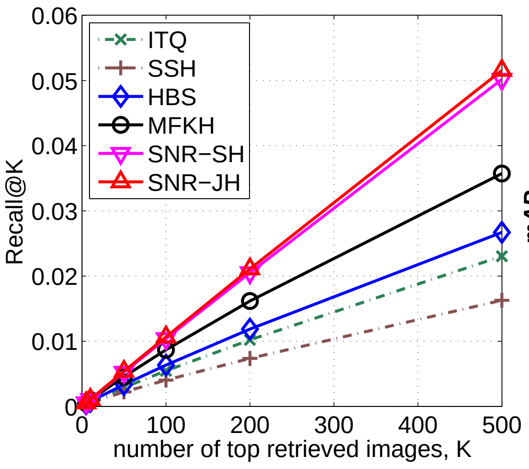
In summary, we have proposed SNR-JH and SNR-SH, two multi-feature hashing methods based on SNR maximization. SNR-JH jointly learns projection directions from all features, while SNR-SH selects projection directions from a pool of candidate projections generated from each individual feature. Despite the simple linear model (4.3) and the simple training pro-



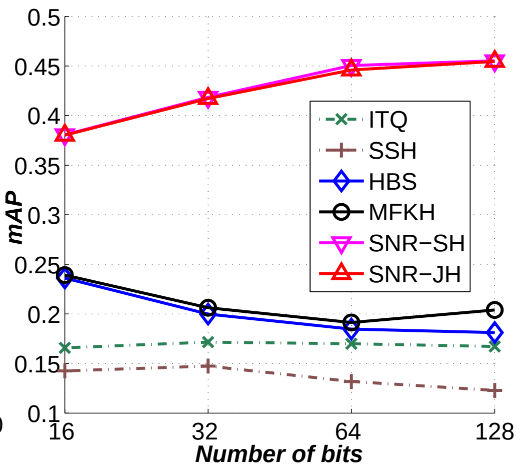
(a) 16 bits.



(b) 32 bits



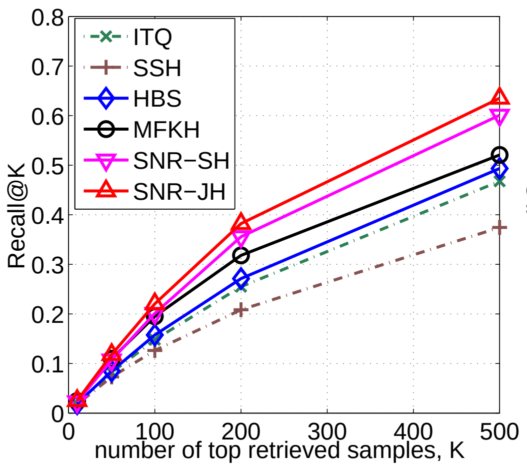
(c) 128 bits



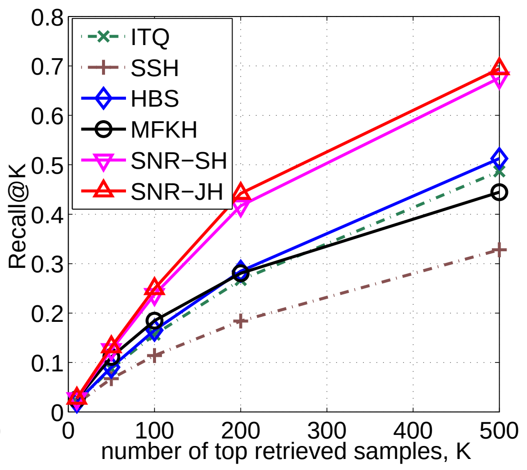
(d) mAP

Figure 6.5: Comparison with state-of-the-art methods on CIFAR-10 dataset.

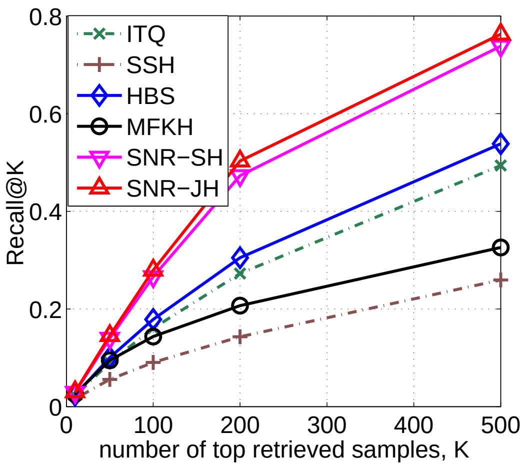
cedure (solving generalized eigenproblems and parameter-free), SNR-JH and SNR-SH have demonstrated superior retrieval performance over MFKH and HBS on five benchmark datasets and excellent classification accuracy on the MNIST digit dataset.



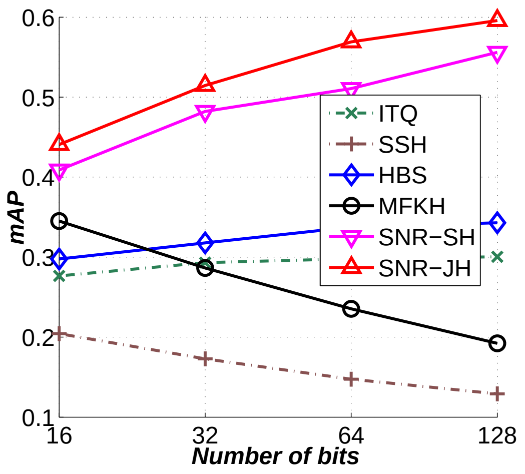
(a) 16 bits.



(b) 32 bits

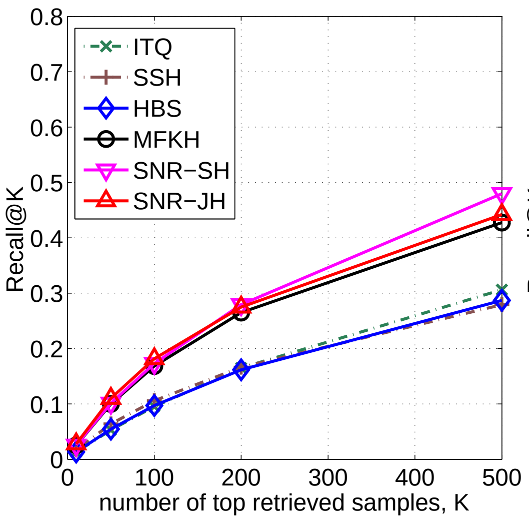


(c) 128 bits

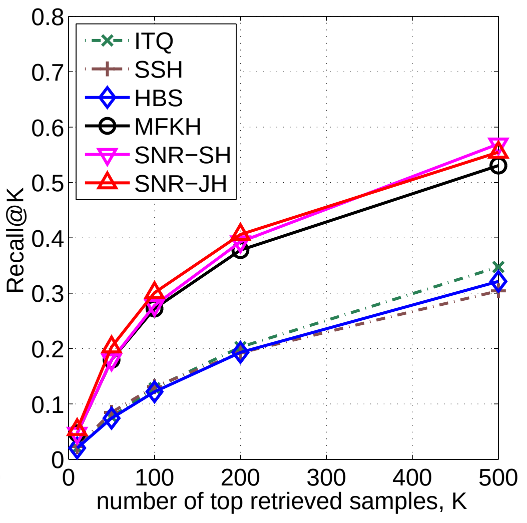


(d) mAP

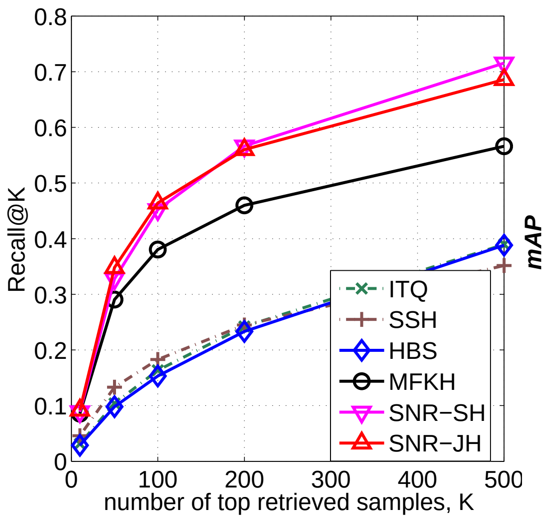
Figure 6.6: Comparison with state-of-the-art methods on Scene-15 dataset.



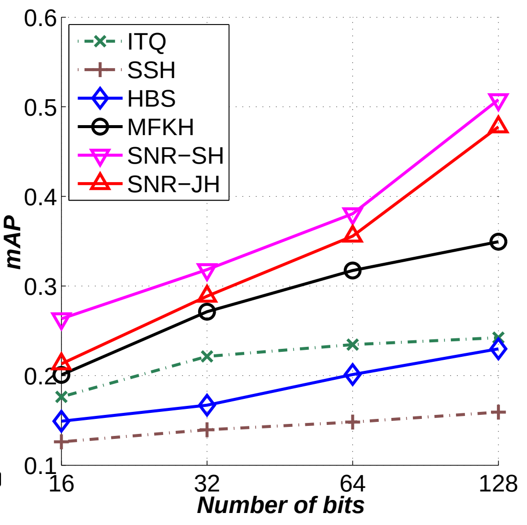
(a) 16 bits.



(b) 32 bits



(c) 128 bits



(d) mAP

Figure 6.7: Comparison with state-of-the-art methods on Caltech-101 dataset.

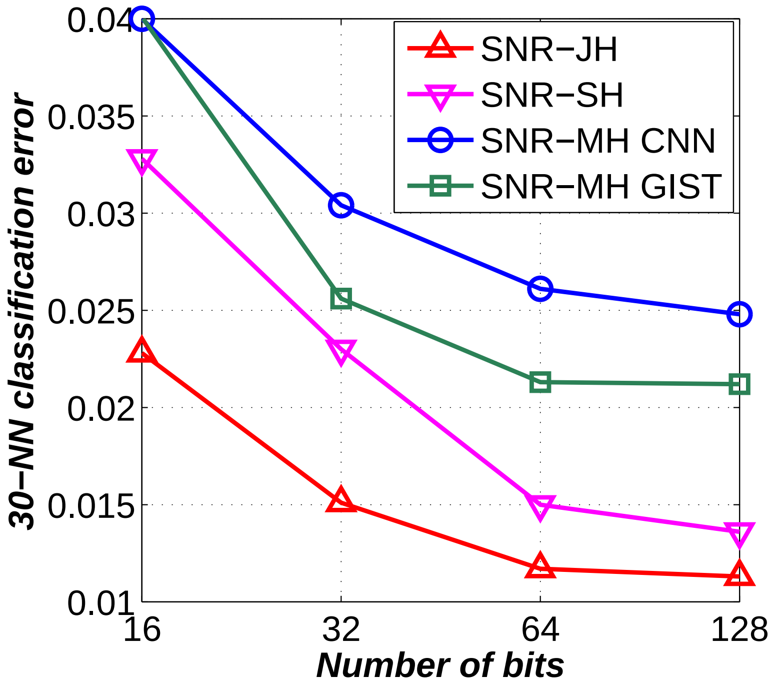


Figure 6.8: Classification performance of multi-feature SNR-JH and SNR-SH, and their single-feature counterparts SNR-MH on the MNIST digit dataset.

CHAPTER 7

HASHING REVISITED: OBSERVATIONS AND OPEN PROBLEMS

When semantic labels are used as the groundtruth, we have observed that nearest neighbor (NN) search using the Hamming metric in Hamming space can perform better than NN search using the Euclidean metric in feature space. This behavior has also been observed by others [7, 8]. It seems that we not only save storage space and increase search speed but also improve performance by hashing. Often the performance improvement is viewed as a pleasant surprise. However, since both the Hamming metric and the Euclidean metric are suboptimal decoding metrics (the optimal likelihood "metric" requires knowledge of true data distribution, which is not available here), there is no reason one should be better than the other. In the first part of this chapter, we empirically compare the Hamming and Euclidean metrics under the SNR maximization framework, and demonstrate that in the low-SNR regime the Hamming metric decisively outperforms the Euclidean metric.

Bit independence is a desired property of many hashing algorithms, such as SNR-MH and algorithms from [5, 12, 65, 79], while others generate correlated bits [9, 6, 22, 86]. Independent bits result in more compact binary representations. For instance, we used regularizers to penalize hash functions generating highly correlated bits in regularized Adaboost, and obtained significant performance gain over the SPB algorithm where temporal correlation is ignored. One may argue that correlated bits can be further compressed by entropy coding, and thus correlated bits should not be avoided. However, entropy coding will result in variable code length and require more complicated decoding schemes than fixed-length codes. Therefore, we will restrict our discussion to fixed code length without entropy coding. On the other hand, it is often difficult to generate equally good independent bits. Under the unsupervised setting, it has been observed that data distributions are concentrated in a few high-variance projections [6, 22]. Under the supervised setting, we have also noticed that the number of high-SNR projections is limited, and bits generated from subsequent uncorrelated low-SNR projections may deteriorate performance. In Chapter 5, we have proposed a multi-bit algorithm SNR-MBH to keep performance improving with longer code length. In the second part of this chapter, we consider a different approach to tackle the low-SNR projections. We show that balancing the SNR among projections, similarly to the ITQ variance balance approach, can lead to better performance. Moreover, we can clearly demonstrate the trade-off between the correlation among projections and the distribution of the SNRs at these projections.

7.1 Hamming Metric versus Euclidean Metric

In Chapter 4, we proposed an SNR maximization procedure to learn K projection directions $W = \{w_1, \ldots, w_K\} \in \mathbb{R}^{D\times K}$ and binarize the projected data to hash codes: $\mathbf{f} = \operatorname{sgn}(W^T\mathbf{x}) \in \{\pm 1\}^K$ and $\mathbf{g} = \operatorname{sgn}(W^T\mathbf{y}) \in \{\pm 1\}^K$. In this section, we empirically compare the retrieval performance for quantized hash codes and unquantized feature projections under the Gaussian model.

We assume dissimilar $\mathbf{X}$ and $\mathbf{Y}$ are i.i.d. Gaussian random vectors, and similar $\mathbf{X}$ and $\mathbf{Y}$ follow the relation of (4.3). To make the analysis easier, we further assume $\mathrm{SNR}_k = \sigma_x^2/\sigma_z^2, k = 1, \ldots, K$. Denote by $\widetilde{\mathbf{x}} = W^T\mathbf{x} \in \mathbb{R}^K$ and $\widetilde{\mathbf{y}} = W^T\mathbf{y} \in \mathbb{R}^K$ the projected feature vectors. We compare the performance of the Euclidean rule

$$d_E(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) \underset{D}{\overset{S}{\lessgtr}} \delta, \tag{7.1}$$

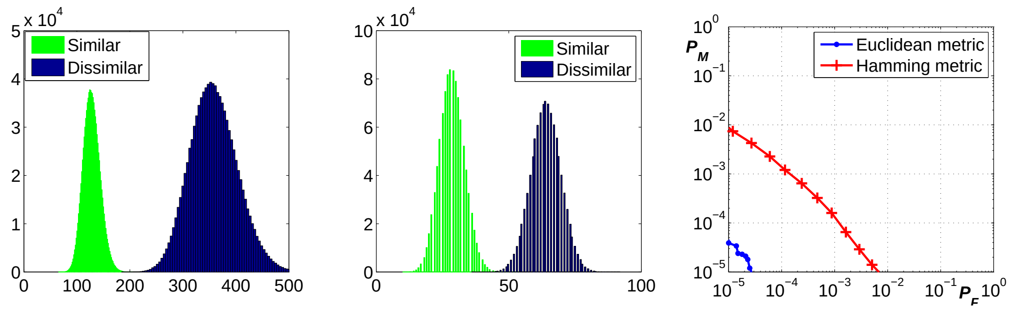
where d_E denotes the squared Euclidean distance, and the Hamming rule

$$d_H(\mathbf{f}, \mathbf{g}) \underset{D}{\overset{S}{\lessgtr}} \tau. \tag{7.2}$$

As the SNR maximization procedure normalizes the noise power to unity, i.e., $\sigma_z^2 = 1$, the only free parameter in our simulations is the SNR. Varying the values of the SNR, we obtain very different conclusions. When SNR is high, the Euclidean metric performs better than the Hamming metric (see Fig. 7.1c). As SNR decreases, histogram separation narrows and performance drops for both metrics. However, the performance drops much more drastically with the Euclidean metric (see Fig. 7.2c and Fig. 7.3c). This behavior can be explained by examining the distributions of $d_E(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}})$ and $d_H(\mathbf{F}, \mathbf{G})$.

Define random variables T_E^S and T_E^D as the squared Euclidean distance between similar and dissimilar $\mathbf{X}$ and $\mathbf{Y}$ respectively, and T_H^S and T_H^D as the Hamming distance between similar and dissimilar $\mathbf{F}$ and $\mathbf{G}$ respectively. Under the Gaussian model, it can be easily shown that $T_E^S \sim \sigma_z^2 \chi_K^2$ and $T_E^D \sim 2\sigma_x^2 \chi_K^2$, where χ_K^2 is the chi-squared distribution with K degrees of freedom. Also, $T_H^D \sim \mathrm{Bi}(K, 1/2)$ and $T_H^S \sim \mathrm{Bi}(K, p)$, where p is a monotonically decreasing function of SNR as shown in (4.14).

Note that when $\sigma_x^2 = 0.5\sigma_z^2$, T_E^S and T_E^D have the same distribution, hence the histograms of T_E^S and T_E^D are indistinguishable. However, p is well below 0.5 for $SNR = 0.5$, causing the histograms of T_H^S and T_H^D to still be well separated. As shown in Fig. 7.4a, p is robust to changes in SNR. It is therefore not surprising that the Hamming metric performs better than the Euclidean metric at low-SNR regime. In Fig. 7.3, we show results for $SNR = 0.6$. The histograms of the Hamming distance are much better separated than those of the squared Euclidean distance, resulting in orders of magnitude performance gap in ROC curves.



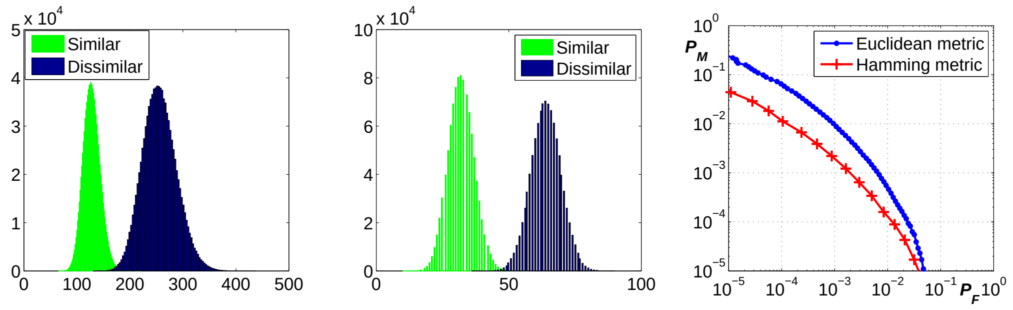
(a) Histograms of $d_E(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ (b) Histograms of $d_H(\mathbf{f}, \mathbf{g})$ (c) ROC curves

Figure 7.1: Results for SNR = 1.4.

Since histogram separation is a strong indicator of the performance, we use a Bhattachayya distance approximation [87]

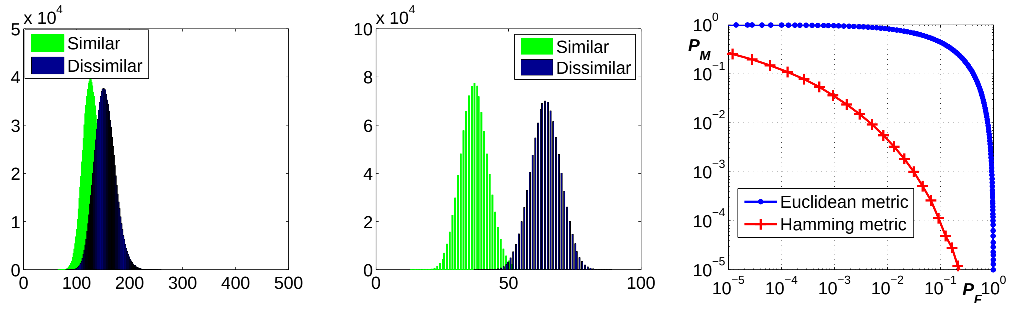
$$d_B(p_1, p_2) = \frac{1}{4} \ln \left(\frac{1}{4} \left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\sigma_2^2}{\sigma_1^2} + 2 \right) \right) + \frac{1}{4} \left(\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \right), \tag{7.3}$$

where (μ_1, σ_1^2) and (μ_2, σ_2^2) are the mean and variance of distribution p_1 and p_2 respectively, to measure the similarity between histograms (T_E^S, T_E^D) and



(a) Histograms of $d_E(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ (b) Histograms of $d_H(\mathbf{f}, \mathbf{g})$ (c) ROC curves

Figure 7.2: Results for SNR = 1.



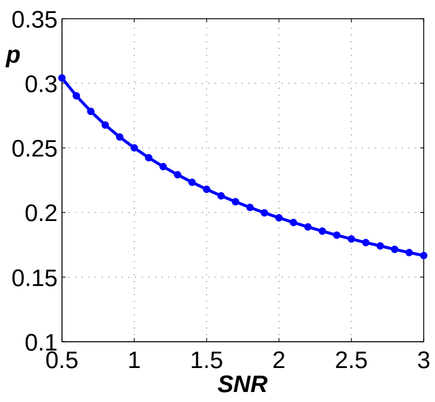
(a) Histograms of $d_E(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ (b) Histograms of $d_H(\mathbf{f}, \mathbf{g})$ (c) ROC curves

Figure 7.3: Results for SNR = 0.6.

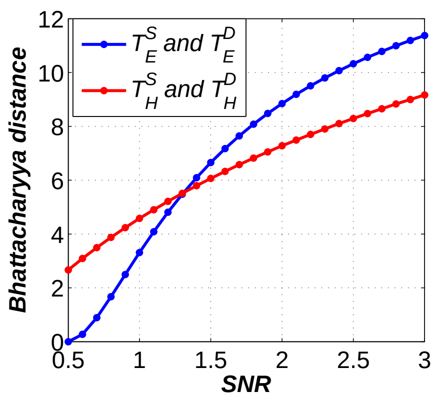
(T_H^S, T_H^D) at different SNR.[1] As shown in Fig. 7.4b, the Euclidean metric has larger Bhattachayya distances in the high-SNR regime, while the Hamming metric dominates in the low-SNR regime.

In this section, we have clearly shown that Hamming metric on binary hash codes can perform better than Euclidean metric on feature vectors. Though we only showed this under the Gaussian model, we would not be surprised to see this behavior over other datasets. Therefore, supervised hashing, where semantic labels are the groundtruth, should not be considered a tool for fast ANN search. In fact, supervised hashing could perform even better than exact NN search!

[1]This assumes Gaussian distributions. As $K = 128$ is relatively large, we approximate the chi-squared and binomial distributions with Gaussian distributions to obtain the rough results shown in Fig. 7.4b.



(a) p versus SNR.



(b) d_B versus SNR.

Figure 7.4: Results for various values of SNR.

7.2 SNR Distribution versus Projection Correlation

In Section 5.1, we have demonstrated that the optimal decision rule for hashing applications can be expressed in terms of the weighted Hamming distance (5.1) under the Gaussian model, where the weight for the k-th bit solely depends on SNR_k. However, the Hamming metric implicitly assigns the same weight to each bit, which causes performance to deteriorate when low-SNR projections are used. To enjoy the simplicity and fast search speed of the Hamming metric, we consider a balancing strategy that forces $\{\text{SNR}_k\}_{k=1}^K$ to be more evenly distributed at different projections.

Denoting by $W = \{w_1, \dots, w_K\} \in \mathbb{R}^{D\times K}$ the optimal projections from solving (4.15), we have $W^T C_Z W = I$ and $W^T C_X W$ is a diagonal matrix with diagonal entries $[\text{SNR}_1, \dots, \text{SNR}_K]$. Thus,

$$\sum_{k=1}^{K} \text{SNR}_k = \text{Tr}\left(W^T C_X W\right). \tag{7.4}$$

To balance the SNRs, we use orthogonal transformations. Multiplying W with any orthogonal $K \times K$ matrix R does not change the total SNR, i.e., $R^T W^T C_Z W R = I$ and $\text{Tr}\left(R^T W^T C_X W R\right) = \text{Tr}\left(W^T C_X W\right) = \sum_{k=1}^K \text{SNR}_k$. Ideally, we would like to find an orthogonal matrix R such that SNR_k's are uniformly distributed across different projections while keeping projections uncorrelated, i.e., $R^T W^T C_X W R = \bar{\gamma} I$ where

$$\bar{\gamma} = \frac{1}{K} \sum_{k=1}^{K} \mathrm{SNR}_k \tag{7.5}$$

is the average SNR. Unfortunately, as shown next, no such R exists! Therefore, balancing SNRs will inevitably introduce correlation among projections.

Denote by $\Lambda = W^T C_X W$ the transformed feature covariance matrix. Note that Λ is a diagonal matrix with diagonal entries $\Lambda_{kk} = \mathrm{SNR}_k, k = 1, \ldots, K$. To measure the difference between $R^T \Lambda R$ and $\bar{\gamma} I$, we have

$$\begin{aligned} J(R) &= ||R^T \Lambda R - \bar{\gamma} I||_F^2 \\ &= \mathrm{Tr}\left((R^T \Lambda R - \bar{\gamma} I)^T (R^T \Lambda R - \bar{\gamma} I)\right) \\ &= \mathrm{Tr}\left(\Lambda^2 - 2\bar{\gamma}\Lambda + \bar{\gamma}^2 I\right) \\ &= \sum_{k=1}^{K} \left(\mathrm{SNR}_k - \bar{\gamma}\right)^2, \end{aligned}$$

which does not depend on R. Unless $\Lambda = \bar{\gamma} I$, $J(R)$ is a positive constant.

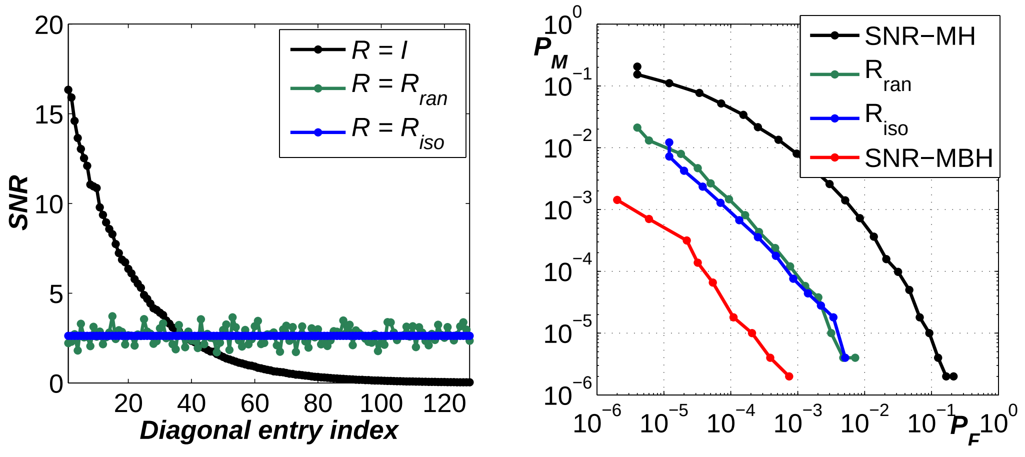
In the rest of this subsection, we will examine two heuristic balancing strategies, and compare them with SNR-MH and SNR-MBH. Hopefully, we can shed some light on this problem.

For a random orthogonal matrix R_{ran}, $R_{\mathrm{ran}}^T \Lambda R_{\mathrm{ran}}$ will have nonzero off-diagonal entries, i.e., nonzero correlations between transformed feature vectors. Since $J(R)$ is a constant, the diagonal entries of $R_{\mathrm{ran}}^T \Lambda R_{\mathrm{ran}}$ will be more evenly distributed with values not far from $\bar{\gamma}$.

From the Shur-Horn Lemma [88], there exists an orthogonal matrix R_{iso} such that diagonal entries of $R_{\mathrm{iso}}^T \Lambda R_{\mathrm{iso}}$ all equal to $\bar{\gamma}$. Such R_{iso} can be obtained by the isotropic hashing procedure [89]. As diagonal entries of $R_{\mathrm{iso}}^T \Lambda R_{\mathrm{iso}}$ and $\bar{\gamma} I$ are equal, the sum of the squared off-diagonal entries attains the maximal value $\sum_{k=1}^{K} \left(\mathrm{SNR}_k - \bar{\gamma}\right)^2$, and thus R_{iso} creates the most correlated feature vectors in terms of the average absolute correlation.

We use the same synthetic dataset as in Chapter 5 and show results for different orthogonal matrices in Fig. 7.5. In Fig. 7.5a, we see that SNRs from the SNR maximization are distributed very unevenly. With a random rotation, SNRs are close to uniform. As R_{iso} forces isotropic diagonal entries, all SNRs are equal. Figure 7.5b compares ROC curves among different schemes. As explained in Section 5.1, SNR-MH suffers from low-SNR projections. Though R_{ran} and R_{iso} inevitably create dependency among projected data, we see a significant performance improvement from SNR-MH because of the more balanced SNRs.

However, balancing SNRs by R_{ran} or R_{iso} still does not bring ROC curves near to that of the SNR-MBH algorithm. So far, there is no systematic way of choosing an orthogonal matrix R that achieves the optimal trade-off between balancing SNRs and controlling correlation. We believe this is an important problem in hashing which will require more investigation in the future.



(a) Diagonal entries of $R^T \Lambda R$.   (b) ROC curves, 128 bits.

Figure 7.5: Results on synthetic dataset with different orthogonal transformations.

CHAPTER 8

CONCLUSIONS

8.1 Summary of Contributions

The main contributions of this dissertation are as follows:

1. Regularized Adaboost, presented in Chapter 3, is a fairly general framework for identification of time-varying content. Audio and video content ID systems use substantial overlapping of slices to mitigate misalignment during identification [30, 31, 32]. Hence fingerprinting algorithms such as SPB [28, 29] produce highly correlated fingerprints. While some correlation in fingerprints is useful to combat misalignment, information-theoretic analysis and real world experiments show that too much correlation is undesirable. Our proposed fingerprinting algorithm is based on the boosting framework and uses a regularization term to control the amount of fingerprint correlation and improve content ID performance. We have proposed a mutual information regularizer (MIR) and an average correlation coefficient regularizer (ACCR), both of which are easy to compute and can capture the filter's ability to decorrelate overlapping slices. Significant performance gains over SPB have been demonstrated for audio, video and RGB-D video content ID systems.

2. SNR maximization hashing (SNR-MH), presented in Chapter 4, is a simple and powerful hashing framework. We have shown that the hash bits generated from SNR maximization projections minimize the hashing error probability under a Gaussian model for the underlying signals. Despite the simple linear model (4.3) and the simple training procedure (solving generalized eigenproblems and parameter-free), SNR-MH exhibits excellent retrieval performance on both synthetic and various

real datasets.

3. The multi-bit hashing of Chapter 5 (SNR-MBH) extends SNR-MH to learn longer hash codes when high-SNR projections are limited. SNR-MBH is an automatic procedure that determines the number of available high-SNR projections, the number of bits for each projection, and the positions of quantization thresholds. SNR-MBH not only demonstrates superior retrieval performance over many other hashing algorithms, but also excellent classification performance with a simple K-NN classifier.

4. SNR joint hashing (SNR-JH) and SNR selection hashing (SNR-SH) of Chapter 6 are two multi-feature hashing algorithms based on SNR maximization. SNR-JH jointly learns projection directions from all features, while SNR-SH selects projection directions from a pool of candidate projections generated from each individual feature. Both SNR-JH and SNR-SH consider the different statistical properties from different features and exploit the complementary information between features. Both SNR-JH and SNR-SH perform favorably compared to other state-of-the-art multi-feature hashing algorithms on five benchmark datasets.

5. In Chapter 7, we present two observations within the SNR maximization framework. The first is that in the low-SNR regime the Hamming metric is decisively better than the Euclidean metric, when semantic label is used as the groundtruth. The second concerns the trade-off between SNR distribution and projection correlation. Though these two observations are only demonstrated in the SNR maximization framework, I believe they can be generalized to other settings and are worth further investigation.

8.2 Future Directions

Besides the open problems in Chapter 7, I would like to discuss two additional directions for future research before concluding this dissertation.

1. **Kernelized SNR maximization hashing**: One of the motivations

of SNR-MH is its optimality under the linear Gaussian model of (4.3). However, the linear model may be too restrictive for the given input feature vector. Potentially, a nonlinear transformation of the input feature space could make the noise model linear in the transformed high-dimensional (possibly infinite dimensional) feature space. To avoid explicit learning in high-dimensional space, kernel methods have been successfully applied to many learning problems, such as support vector machines and kernelized locality sensitive hashing [3, 4]. Therefore, constructing a kernelized SNR-MH algorithm may be helpful in increasing the capacity of SNR-MH.

2. **Combining SNR maximization and deep neural networks:** Among the many image features we have explored in our experiments, the CNN feature constructed from a deep convolutional neural network (CNN) [74] is certainly the most powerful one. Applying SNR maximization hashing on top of the CNN feature, we have achieved start-of-the-art retrieval and classification performances on many image datasets. It might be promising to combine the two in a unified framework. For instance, one might want to combine the deep CNN architecture and the supervised SNR metric to learn binary codes from large amounts of labeled or weakly labeled data.

APPENDIX A

RELATION BETWEEN P_{FN} AND P_{MISS}

To derive the link between P_{FN} and P_{miss} for the two decoders in Def. 2 and Def. 3, we express P_{FN} and P_{miss} in terms of the decision regions.

For the single-output decoder ψ, the decision region R_m^ψ for $m \in \{1, 2, \ldots, M\}$ is given by

$$R_m^\psi = \{\mathbf{g} : d^*(\mathbf{f}(m), \mathbf{g}) < \tau \text{ and } d^*(\mathbf{f}(m), \mathbf{g}) < d^*(\mathbf{f}(m'), \mathbf{g}), \forall m' \neq m\}. \quad \text{(A.1)}$$

For the variable-size list decoder $\mathcal{L}$, the decision region $R_m^{\mathcal{L}}$ for $m \in \{1, 2, \ldots, M\}$ is given by

$$R_m^{\mathcal{L}} = \{\mathbf{g} : d^*(\mathbf{f}(m), \mathbf{g}) < \tau\}. \quad \text{(A.2)}$$

While $\{R_m^\psi\}$ are disjoint sets, $\{R_m^{\mathcal{L}}\}$ are generally overlapping. It follows from (3.2) and (3.3) that

$$P_{FN} = \frac{1}{M} \sum_{m=1}^{M} \sum_{\mathbf{g} \notin R_m^\psi} Pr[\mathbf{g}|H_m], \quad \text{(A.3)}$$

$$P_{miss} = \frac{1}{M} \sum_{m=1}^{M} \sum_{\mathbf{g} \notin R_m^{\mathcal{L}}} Pr[\mathbf{g}|H_m]. \quad \text{(A.4)}$$

Clearly, we have $R_m^\psi \subseteq R_m^{\mathcal{L}}$. Therefore,

$$P_{miss} \leq P_{FN}. \quad \text{(A.5)}$$

APPENDIX B

EQUIVALENCE BETWEEN SNR MAXIMIZATION (4.15) AND GENERALIZED EIGENPROBLEM (4.16)

Denote by $\gamma(w)$ the objective function of (4.15)

$$\gamma(w) = \frac{w^T C_X w}{w^T C_Z w}, \tag{B.1}$$

where both C_x and C_z are symmetric and C_z is positive-definite. The ratio $\gamma(w)$ is known as the *Rayleigh quotient*. We first show that the critical points of $\gamma(w)$ correspond to the eigensystem of the generalized eigenproblem of (4.16), and then show that these eigenvectors satisfy the constraints of (4.15).

Setting the gradient of γ equal to zero, we have

$$\frac{\partial \gamma}{\partial w} = \frac{2}{w^T C_z w}(C_x w - \gamma C_z w) = 0. \tag{B.2}$$

Since $w^T C_z w > 0$, we have

$$C_x w = \gamma C_z w, \tag{B.3}$$

which is the generalized eigenproblem of (4.16).

Next, we show eigenvectors from (B.3) satisfy the constraints of (4.15), i.e.,

$$\begin{aligned} w_i^T C_x w_j &= 0 \quad \forall i \neq j \\ w_i^T C_z w_j &= 0 \quad \forall i \neq j \\ w_i^T C_z w_i &= 1 \quad \forall i. \end{aligned}$$

Note that $w_i C_z w_i > 0$ can always be forced to be one as γ is invariant to scaling in w_i. For the i-th eigenvalue γ_i and eigenvector w_i, we have

$$C_x w_i = \gamma_i C_z w_i. \tag{B.4}$$

The dot product with another eigenvector w_j gives

$$w_j^T C_x w_i = \gamma_i w_j^T C_z w_i. \tag{B.5}$$

Similarly, we have

$$w_i^T C_x w_j = \gamma_j w_i^T C_z w_j. \tag{B.6}$$

As both C_x and C_z are symmetric, we have $w_j^T C_x w_i = w_i^T C_x w_j$ and $w_j^T C_z w_i = w_i^T C_z w_j$. Therefore,

$$\gamma_i w_i^T C_z w_j = \gamma_j w_i^T C_z w_j \tag{B.7}$$

$$\frac{1}{\gamma_i} w_i^T C_x w_j = \frac{1}{\gamma_j} w_i^T C_x w_j. \tag{B.8}$$

When $\gamma_i \neq \gamma_j$, we have $w_i^T C_x w_j = 0$ and $w_i^T C_z w_j = 0$.

We have shown that the eigenvectors corresponding to the generalized eigenproblem (4.16) are the solution to the SNR maximization problem (4.15).

REFERENCES

[1] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast search in hamming space with multi-index hashing," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2012, pp. 3108–3115.

[2] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *ACM STOC*, 1998, pp. 604–613.

[3] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[4] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2009, pp. 1509–1517.

[5] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2008, pp. 1753–1760.

[6] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11, Washington, DC, USA, June 2011, pp. 817–824.

[7] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *NIPS*, 2012, pp. 1070–1078.

[8] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, Jan. 2012.

[9] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter sensitive hashing," in *In ICCV*, 2003, pp. 750–757.

[10] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *ICML*, 2011, pp. 353–360.

[11] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, July 2009.

[12] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*, San Francisco, CA, June 2010, pp. 3424–3431.

[13] R. Venkatesan, S.-M. Koon, M. H. Jakubowski, and P. Moulin, "Robust image hashing," in *IEEE International Conference on Image Processing: ICIP 2000*, Vancouver (BC), CA, September 2000.

[14] P. Moulin, "Statistical modeling and analysis of content identification," in *Information Theory and Applications Workshop*, San Diego, CA, February 2010.

[15] A. L. Varna and M. Wu, "Modeling and analysis of correlated binary fingerprints for content identification," *IEEE Transactions on Information Forensics and Security*, vol. 6, pp. 1146–1159, 2011.

[16] R. Naini and P. Moulin, "Model-based decoding metrics for content identification," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, March 2012, pp. 1829 –1832.

[17] F. Willems, T. Kalker, S. Baggen, and J. paul Linnartz, "On the capacity of a biometrical identification system," in *In: Proc. of the 2003 IEEE Int. Symp. on Inf. Theory*, Yokohama, Japan, 2003, pp. 8–2.

[18] A. L. Varna, "Multimedia protection using content and embedded fingerprints," Ph.D. dissertation, University of Maryland, 2011.

[19] T. Holotyak, S. Voloshynovskiy, O. J. Koval, and F. Beekhof, "Fast physical object identification based on unclonable features and soft fingerprinting," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 1713–1716.

[20] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry.* ACM, 2004, pp. 253–262.

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

[22] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in *ICML*, Haifa, Israel, June 2010, pp. 1127–1134.

[23] H. Jégou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 33, no. 1, pp. 117–128, jan 2011. [Online]. Available: http://lear.inrialpes.fr/pubs/2011/JDS11

[24] M. Norouzi and D. J. Fleet, “Cartesian k-means,” in *CVPR*, 2013, pp. 3017–3024.

[25] T. Ge, K. He, Q. Ke, and J. S. 0001, “Optimized product quantization for approximate nearest neighbor search,” in *CVPR*, 2013, pp. 2946–2953.

[26] K. He, F. Wen, and J. Sun, “K-means hashing: An affinity-preserving quantization method for learning binary compact codes,” in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2013, pp. 2938–2945.

[27] Y. Ke, D. Hoiem, and R. Sukthankar, “Computer vision for music identification,” in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, Washington, DC, USA, 2005, pp. 597–604.

[28] D. Jang, C. D. Yoo, S. Lee, S. Kim, and T. Kalker, “Pairwise boosted audio fingerprint,” *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 995–1004, Dec. 2009.

[29] S. Lee, C. D. Yoo, and T. Kalker, “Robust video fingerprinting based on symmetric pairwise boosting,” *IEEE Trans. Circ. and Sys. for Video Technol.*, vol. 19, no. 9, pp. 1379–1388, Sep. 2009.

[30] J. Haitsma and T. Kalker, “A highly robust audio fingerprinting system,” in *Proceedings of the International Symposium on Music Information Retrieval*, Paris, France, 2002.

[31] S. Baluja and M. Covell, “Audio fingerprinting: Combining computer vision data stream processing,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, April 2007.

[32] J. Lu, “Video fingerprinting for copy identification: from research to industry applications,” in *SPIE Electric Imaging Symposium on Media Forensics and Security I*, San Jose, CA, USA, 2009.

[33] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, “A review of audio fingerprinting,” *VLSI Signal Processing*, vol. 41, no. 3, pp. 271–284, 2005.

[34] T. M. Cover and J. A. Thomas, *Elements of Information Theory, 2nd Edition (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, July 2006.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer-Verlag, 2001.

[36] J. H. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, pp. 400–407, 1998.

[37] IACC, "http://www-nlpir.nist.gov/projects/tv2012/pastdata/sin.master.shot.reference.iacc.1.c/iacc.1.C.collection.xml."

[38] J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time," in *CVPR*, 2011, pp. 753–760.

[39] GoldWave, "http://www.goldwave.com."

[40] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, "Audio fingerprinting based on normalized spectral subband moments," *IEEE Signal Process. Lett.*, vol. 13, no. 4, pp. 209–212, 2006.

[41] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," in *Machine Learning for Computer Vision*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2013, vol. 411, pp. 49–87.

[42] "http://www.xbox.com/en-US/KINECT."

[43] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, June 2011.

[44] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proceedings of the European Conference on Computer Vision*, Firenze, Italy, October 2012.

[45] C. Wolf, J. Mille, L.E., Lombardi, O. Celiktutan, M. Jiu, M. Baccouche, E. Dellandrea, C. E. Bichot, C. Garcia, and B. Sankur, "The LIRIS human activities dataset and the ICPR 2012 human activities recognition and localization competition," LIRIS Laboratory, Tech. Rep., 2012.

[46] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *IEEE International Conference on Robotics and Automation*, 2011.

[47] B. Ni, G. Wang, and P. Moulin, "RGBD-HuDaAct: A color-depth video database for human daily activity recognition," in *IEEE Workshop on Consumer Depth Cameras for Computer Vision in conjunction with ICCV*, 2011. [Online]. Available: http://www.adsc.illinois.edu/demos.html

[48] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/1437

[49] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, 2004.

[50] A. Srivastava, A. B. Lee, E. P. Simoncelli, and S.-C. Zhu, "On advances in statistical modeling of natural images," *Journal of Mathematical Imaging and Vision*, vol. 18, pp. 17–33, 2003.

[51] E. P. Simoncelli, "Statistical modeling of photographic images," in *Handbook of Image and Video Processing*, A. Bovik, Ed. Academic Press, May 2005, ch. 4.7, pp. 431–441, 2nd edition.

[52] P. Moulin and J. Liu, "Analysis of multiresolution image denoising schemes using Generalized-Gaussian and complexity priors," *IEEE Trans. Info. Theory*, vol. 45, pp. 909–919, 1998.

[53] M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using Generalized Gaussian density and Kullback-Leibler distance," *IEEE Trans. Image Processing*, vol. 11, pp. 146–158, 2002.

[54] A. Saxena, S. H. Chung, and A. Y. Ng, "3-D depth reconstruction from a single still image," *International Journal of Computer Vision (IJCV)*, vol. 76, 2007.

[55] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. R. Dick, "Learning hash functions using column generation," in *ICML*, ser. JMLR Proceedings. JMLR.org, 2013, pp. 142–150.

[56] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *in Proc. NIPS, 2009*, 2009, pp. 1042–1050.

[57] H. Yu and P. Moulin, "Regularized Adaboost for content identification," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[58] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[59] J. R. Barry, D. G. Messerschmitt, and E. A. Lee, *Digital Communication: Third Edition.* Norwell, MA, USA: Kluwer Academic Publishers, 2003.

[60] K. S. Gomadam and S. A. Jafar, "Optimal relay functionality for snr maximization in memoryless relay networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 2, pp. 390–401, 2007.

[61] B. van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Magazine*, vol. 5, pp. 4–24, April 1988.

[62] R. A. Monzingo, R. L. Haupt, and T. W. Miller, *Introduction to Adaptive Arrays.* Institution of Engineering and Technology, 2011.

[63] B. N. Parlett, *The Symmetric Eigenvalue Problem.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[64] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 1360–1365.

[65] X. Liu, J. He, D. Liu, and B. Lang, "Compact kernel hashing with multiple features," in *Proceedings of the 20th ACM International Conference on Multimedia*, New York, NY, USA, 2012, pp. 881–884.

[66] S. M. Samuels, "On the number of successes in independent trials," *The Annals of Mathematical Statistics*, vol. 36, no. 4, pp. 1272–1278, 08 1965.

[67] F. Balado, N. Hurley, E. McCarthy, and G. Silvestre, "Performance analysis of robust audio hashing," *Information Forensics and Security, IEEE Transactions on*, vol. 2, no. 2, pp. 254–266, June 2007.

[68] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[69] H. Yu and P. Moulin, "Regularized Adaboost learning for identification of time-varying content," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1606–1616, 2014.

[70] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2006, pp. 2161–2168.

[71] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *ICCV*, 2003.

[72] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, May 2001.

[73] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[74] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR 2014)*, April 2014.

[75] Y. Fu, L. Cao, G. Guo, and T. S. Huang, "Multiple feature fusion by subspace learning," in *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, Niagara Falls, Canada, 2008, pp. 127–134.

[76] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *Proceedings of the 19th ACM International Conference on Multimedia*, Scottsdale, Arizona, USA, 2011, pp. 423–432.

[77] H. Yu, P. Moulin, and S. Roy, "RGB-D Video Content Identification," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[78] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, 2011, pp. 225–234.

[79] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: A unified solution for selection problems in hashing," in *CVPR*, 2013.

[80] Y. Zhen and D.-Y. Yeung, "Co-regularized hashing for multimodal data," in *NIPS*, 2012, pp. 1385–1393.

[81] D. Zhai, H. Chang, Y. Zhen, X. Liu, X. Chen, and W. Gao, "Parametric local multimodal hashing for cross-view similarity search," in *IJCAI*, 2013.

[82] J. Schmidhuber, J. Masci, M. M. Bronstein, and A. M. Bronstein, "Multimodal similarity-preserving hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 824–830, 2014.

[83] "http://yann.lecun.com/exdb/mnist/."

[84] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR*, 2006.

[85] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories,” *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 59–70, Apr. 2007.

[86] R.-S. Lin, D. A. Ross, and J. Yagnik, “SPEC hashing: Similarity preserving algorithm for entropy-based coding,” in *CVPR*, 2010, pp. 848–854.

[87] G. Coleman and H. C. Andrews, “Image segmentation by clustering,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, May 1979.

[88] A. Horn, “Doubly stochastic matrices and the diagonal of a rotation matrix,” *American Journal of Mathematics*, vol. 76, no. 3, pp. 620–630, Jul. 1954.

[89] W. Kong and W.-J. Li, “Isotropic hashing,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1646–1654.