Similarity Search with Multimodal Data

Zhe Wang

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Kai Li

January 2012

Abstract

Similarity search systems are designed to help people to organize multimedia non-text data and find valuable information. The multimedia data intrinsically has multiple modalities (e.g., visual and audio features from video clips) which can be exploited to construct better search systems. Traditionally, various integration techniques have been used to aggregate multiple modalities. However, such algorithms do not scale well for large datasets. As the multimedia data grows, it is a challenge to build a search system to handle large-scale multimodal data efficiently and provide users with information they need.

The goal of this dissertation is to study how to effectively combine multiple modalities to implement similarity search systems for large datasets. I have carried out my study through three similarity search systems each designed for different application. Each system combines multiple modalities to help users find desired information quickly. With VFerret system, I studied how to combine visual features with audio features for effective personal video search. With Image Spam Detection System, I explored several aggregation methods to integrate multiple image spam filters to detect image spams. With my Product Navigation System, I studied how to combine text search with image similarity search to help user find desired products. This thesis has also studied a rank-based model which helps system designers to construct more efficient large-scale multimodal similarity search systems.

Although the general solution to using multimodal data in a similarity search system is still unknown, this dissertation shows that it is possible to substantially improve search accuracy and efficiency by leveraging domain specific knowledge of multimodal data. The VFerret system improves search accuracy from an average precision of 0.66 to 0.79 by combining visual and audio features. The Image Spam Detection System significantly lowers the false positive rate from a previous result of 1% to 0.001% while maintaining comparable detection rates by combining multiple

image filters intelligently. My Product Navigation System reduces number of user clicks by 60% compared to traditional systems through a new method of combining text search with image similarity search. These results support further adoption and study of multimodal data in similarity search system designs.

Acknowledgements

First and foremost I would like to thank my adviser Kai Li for his encouragement, guidance, experience and support. I am extremely fortunate to work with Kai and have learned a lot from him. Kai showed me not only how to think openly, but also how to execute my ideas practically. It was a pleasure to be his student and to work in his research group.

I am very grateful to Moses Charikar for being my unofficial second adviser. From group discussions to paper deadlines, Moses always contributes insightful ideas and provides great help. His strong theoretical knowledge and creative thinking guide me throughout my graduate study.

I would also like to thank Olga Troyanskaya for being my thesis reader and Jennifer Rexford and Andrea LaPauph for serving in my thesis committee. Also many thanks to Vivek Pai, Jaswinder Singh, Fei-fei Li and Perry Cook for their guidance and support.

I feel lucky to have met so many good friends in Princeton. Very special thanks to Qin Lv, William Josephson, Wei Dong. Collaborating with them has been a truly inspirational experience. Many thanks to Kyoungsoo Park, Matthew Hoffman for several enjoyable works in my early graduate days. It is a great pleasure to work closely with people within my research group: Yungang Bao, Chris Bienia, Jia Deng, Yida Wang and Wendy Zhang. I feel honored to have had opportunity to work with so many brilliant people and friends here in Princeton: Nitin Garg, Jordan Boyd-Garber, Shirley Gaw, Limin Jia, Jia Li, Junwen Lai, Ruoming Pang, Yaoping Ruan, Haakon Ringberg, Grant Wallace, Yong Wang, Dinghao Wu, Wen Xu, Qian Xi, Bangpeng Yao, Harlan Yu, Chi Zhang, Ming Zhang, Yaping Zhu. Also, I owe a special thanks to Melissa Lawson for all her help during my Princeton years.

This work was supported in part by Gigascale Systems Research Center, Google Research Grant, Yahoo Research Grant, Intel Research Council, National Science

Foundation grants CSR-0509447 and CSR-0509402.

Finally, special thanks to my grandma, Ruidi Cheng who helped me to become who I am today. Many thanks to my parents, Chuilin Wang and Xiaoxin Han, for their love, guidance and support and my in-laws, Delong Shang and Ying Ma, for their support. And to my love, Hongyu Shang, for her endless support and encouragement; to my kids, Hallie and Derek for bringing joy to my life.

The materials of Chapter 2, 3 and 5 have been used in the following publications and presented in the corresponding conferences:

[Chapter 2] Zhe Wang, Matthew Hoffman, Perry Cook and Kai Li. VFerret: Content-Based Similarity Search Tool for Continuous Archived Video. In *Proceedings of the 3rd ACM Workshop on Capture, Archival and Retrieval of Personal Experiences (CARPE). Santa Barbara, California, USA. October 2006*

[Chapter 3] Zhe Wang, William Josephson, Qin Lv, Moses Charikar and Kai Li. Filtering image spam with near-duplicate detection. In *Proceedings of the 4th Conference on Email and Anti-Spam (CEAS). Mountain View, CA, USA. August 2007*

[Chapter 5] Zhe Wang, Wei Dong, William Josephson, Qin Lv, Moses Charikar, Kai Li. Sizing Sketches: A Rank-Based Analysis for Similarity Search. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. San Diego, CA, USA. June 2007*

Contents

Chapter 1

Introduction

1.1 Similarity search with multimodal data

We live in an exploding digital universe and the amount of data has grown exponentially. According to a 2011 study by IDC [57], the digital universe, or the amount of the digital information created and replicated, is estimated to be 1.2 zettabytes(10^{21} bytes) in 2010 and to grow to 1.8 zettabytes in 2011. The dominating data types are scientific sensory data and multimedia data such as image, audio and video. The grand challenge is to find valuable information from such massive amounts of data quickly. For example, search systems for scientific data can help make new scientific discoveries, search systems for image and video data can help people better organize their data and find desired products faster, and search systems for commercial TV and broadcast audio advertisements can help market analysis and improve business decision making.

There have been multiple methods proposed to search for information from multimedia data. In the past, users tried to index such data by manually labelling it with tags and then searching for it with keywords. But this is not a scalable solution when the amount of such data grows quickly. As an example, Table 1.1 shows

Website	Time Frame	New Contents	Data Size	Source
Twitter (text msg)	Feb,2011	140M tweets/day	20GB/day	[118]
Youtube (video)	Mar,2010	24 hours/min	6TB/day	[136]
Facebook (image)	2011 new year weekend	750M photos	60TB/day	[40]

Table 1.1: Data size: comparison of text data vs non-text data

that the amount of non-text data generated by users greatly exceeds traditional text data. In recent years, the focus has shifted towards building content-based similarity search system. The idea is to allow users to quickly find contents that are similar to those they already found via other methods. This would greatly speed up the search process on unlabeled data.

Most of the research on existing similarity search systems has focused on finding the best features for the similarity search task. But multimedia data intrinsically contains multiple modalities. For example, video data contains both visual and audio information, and image data contains color, shape and texture information, etc. Two important open questions in this realm are:

- Is combining multiple features always beneficial, and
- How should one effectively combine features in general?

This thesis studies real system issues on how to integrate multiple features effectively for large-scale datasets.

The standard method to construct a similarity search system with multimodal data is to use "K-Nearest-Neighbor"(KNN) search method. Here, we define distances between pairs of objects; given a query object, the search system returns the K data objects that are nearest to the query object according to the distance function. In order to define distances, we first convert both the query object and the data objects into different types of feature vectors, with one type of feature vector for each modality. The distance between a pair of objects is obtained by summing up

feature distances for each modality. This gives rise to a KNN search problem on a high dimensional space.

Equation 1.1 below shows how multiple features are typically aggregated. Here, we have M modalities. Variable w_m is the feature weight for each modality. Data object $Data_m$ is represented by a D-dimensional real-valued vector. $d(X, Y)$ is a domain specific distance function, and it is usually modeled by one of the ℓ_p norms. We want to find K data objects with the smallest total distance to the query object: $Dist(Query, Data)$.

$$\begin{aligned} Dist(Query, Data) &= \sum_{1 \le m \le M} (w_m * d(Query_m, Data_m)) \\ d(X, Y) &= \left(\sum_{1 \le i \le D} (X_i - Y_i)^p \right)^{1/p} \end{aligned}$$

Figure 1.1 shows the typical architecture of a similarity search system with multimodal data. The example depicts a similarity search engine for video data. The system works in two phases. The first phase is an offline data processing stage where all video data that needs to be indexed is processed offline to speed up online searches later. The second phase is the online query stage, where the user submits a query video clip and the system returns all the similar video clips.

For the offline data processing stage, it is typical to first extract certain features from the multimedia data that can be used to best "describe" the contents. That is, if two pieces of video clips are considered to be similar, the features extracted from them should also be close to each other in the high dimensional space according to certain distance measures. Typically, the extracted feature is a high dimensional vector, such as a color histogram. The distance measure here can be a simple L_1

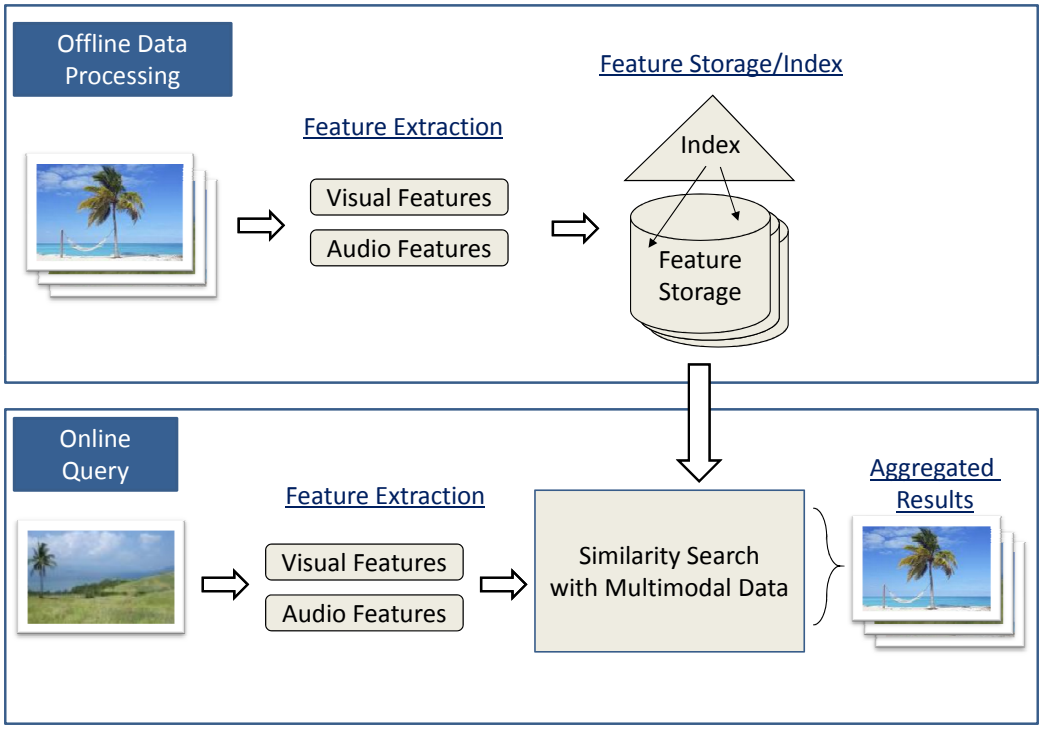


Figure 1.1: Similarity Search System with Multimodal Data

distance. Alternately, the extracted feature could be a set of feature vectors, such as a collection of region based local features as described in Lv's paper [79]. Here, one possible distance measure is the Earth Mover Distance [93] which defines a distance between two sets of feature vectors.

Since there are different perspectives on defining similarity, we can have multiple kinds of features describing the same content. In the example illustrated in Figure 1.1, we can extract visual features such as color and shape from the image of video and audio features from the sound track of the video. These multiple kinds of features form the multiple modalities of the video data. If combined properly, they can potentially improve the final similarity search accuracy.

The second step in the offline processing stage is to insert all features into the feature database and build an index for fast retrieval. The traditional indexing methods are based on space partitioning techniques such as R-tree [55], K-D tree [12] and

SR-tree [65]. Recently, several indexing methods based on locality sensitive hashing (LSH) [60, 50, 30, 87] have been proposed for approximate KNN search. By relaxing the search criteria to conduct approximate KNN search rather than exact KNN search, it is possible to significantly speed up the search process while losing only a small percentage of precision.

For the online query stage, the user will submit a video clip as the query to initiate the similarity search. The query video clip will go through the same feature extraction process as in the offline processing stage. The extracted feature vectors are compared with the stored ones in the feature databases to find similar objects. This kind of KNN similarity search will generate a list of candidate video clips that are closest to the query video clip in the feature space. For objects with multiple kinds of features, it is typical to assign different weights to each feature and calculate overall object distance as a combination of individual feature distances.

1.2 Previous Work

While domain experts on multimedia data have done significant research on searching such data, they still have difficulties in handling large-scale datasets. For example, many complicated algorithms work well with tens of thousands of objects, but are not capable of handling datasets with billions of objects. From a system designer's perspective, I am interested in investigating practical methods for performing similarity searches on large-scale datasets and effectively choosing multiple simple features to achieve the best possible accuracy.

There are various lines of research involving similarity search with large-scale multimodal data. We can categorize them into theoretical studies and system implementations, and further split each into research working with multimodal data and research focusing on large scale datasets. I have summarized these lines of research

in the following four sections:

1.2.1 Theoretical Results on Multimodal Data

In theoretical studies of integration methods with various multimodal datasets, researchers have investigated different methods of combining multiple features to achieve higher accuracy for similarity search. Since features are designed with different goals in mind, they could be focusing on different aspects of the data. As shown in Figure 1.1, the features can be designed to capture different properties of the objects such as color, shape and texture. Modern computer vision algorithm can aggregate dozens of features to boost the accuracy of the search [48].

There are two main approaches to build systems to optimally combine multiple features for better accuracy. The first approach uses learning based fusion techniques: Verlinde compared different decision fusion techniques in designing a multimodal biometric identity verification system [122]. The paper demonstrate the strength of the "logistic regression" method over other fusion methods such as "Maximum Likelihood," "Maximum A-Posteriori Probability," etc. Recently, researchers have studied more advanced learning methods such as SVM based "Super Kernel Fusion" [135] and "Multiple Kernel Learning" [8], as well as boosting based methods such as "CG-Boosting" [14], and "LP-Beta" [48].

The second approach uses non-learning based methods. The search system adopts a traditional probabilistic Bayesian framework as described by Kittler [69]. It uses various rules such as sum rule, product rule and min/max rule to aggregate results from multiple features in the KNN similarity search framework. It has been reported in recent papers [15, 11] to be as effective as sophisticated machine learning methods in large-scale image similarity search.

1.2.2 Similarity Search Systems with Multimodal Data

Researchers have designed various experimental similarity search systems with multimodal data in different research fields such as biometric verifications, image similarity search, video similarity search, etc.

Biometric verification research [41, 104, 29] has shown performance improvement from combining multiple biometric characteristics such as fingerprints, 2D/3D face traits, signatures, hand geometry, etc. Ross provides a survey [92] on information fusion in biometrics.

In providing image similarity search systems, various researchers [135, 8, 48] have merged multiple image features invented by prior research to generate better classification results. The fusion methods used include SVM based approach [135], Multiple Kernel Learning [8], boosting based approaches [14, 48], and KNN based approaches [15, 11].

Researchers have also taken advantage of multiple modalities exhibited in video data when building video content-based search systems. The video data inherently comes with visual and audio content. Moreover, it is often possible to extract other features such as captions present in commercial video, screen text information present in the news video [76], tags and concepts created by online users [74, 130], click through information collected by online video servers [129] and context information found in online social network [53]. Recent progress has been described in two survey papers: "Surveys on Multimodal Video Indexing" [106] and "Multimodal Video Representation for Semantic Analysis" [5].

1.2.3 Theoretical Work on Dimension Reduction

In theoretical studies of algorithmic approaches to large-scale datasets, recent research has both studied and employed compact data representations (sketches) of high dimensional data, dimension reduction techniques, algorithms for building indexes for

high dimensional large-scale datasets and streaming algorithms.

A compact data representation (sketch) is helpful in dealing with large-scale high dimensional datasets. It reduces the storage requirement for the feature data and improves search speed. A sketch is a compact representation for the original high dimensional feature vector. Moreover, it is designed such that the distance between the high dimensional feature data can be approximated by the sketch distance. The sketch construction is dependent on the distance function used on the feature data. Charikar [23] showed in 2002 how sketch constructions can be derived from rounding techniques used in approximation algorithms. Later in 2004, he also developed a new sketching technique for estimating the Earth Mover's Distance in [80]. Such sketching techniques reduce of the feature data storage requirement by almost an order of magnitude while still retaining about 90% accuracy in the similarity search [126].

Researchers have investigated dimension reduction techniques to reduce the dimensionality of high dimensional feature data while still maintaining the quality of features in similarity search. The traditional dimension reduction techniques include Principle Component Analysis (PCA) and Independent Component Analysis (ICA) which both try to summarize the data with fewer dimensions. Indyk and Motwani introduced Locality-Sensitive Hashing (LSH) [60] to perform probabilistic dimension reduction on high dimensional data. They designed a hash function such that the probability of hash collision is higher for objects that are closer in the high dimensional space. LSH can be used to construct an LSH index as described in the next paragraph.

For similarity search, the traditional indexing method does not work efficiently since it is designed to search for exact matches. New indexing techniques have been designed to speed up similarity search. Techniques such as R-tree [55], K-D tree [12] and S-R tree [65] have been proposed, but their performances suffer when the data dimensionality increases. Recently, the LSH indexing method based on p-stable dis-

tribution [30] has been studied extensively to index high dimensional data. Further improvements such as multi-probe LSH [81] and "A posteriori multi-probe LSH" [63] have tried to intelligently probe multiple buckets of LSH hash tables to reduce the space requirements of the LSH index.

The streaming algorithm processes the large-scale dataset in a different way. Initially formalized by Alon in his paper "The Space Complexity of Approximating the Frequency Moments" [2], streaming algorithms try to produce approximate answers by processing data stream only once or with a few passes. Typical applications of streaming algorithms include finding frequent items in data streams [24], clustering datasets [25], counting distinct elements [10] and estimating entropy of a stream [22].

1.2.4 Other Similarity Search Systems

Similarity search systems for large-scale high dimensional data have been implemented in various research areas. The most popular application of similarity search is image similarity search. There were early research prototypes of image similarity search in 1990s such as: QBIC [42], Windsurf [7], SIMPLIcity [125]. Recent advances in indexing techniques, as well as growing image data availability, enable image similarity search to be performed on much larger datasets. The Tiny Image project crawled and indexed 80 million images [117]; CoPhIR project collected 100 million images [16]. For more research projects, Datta has a comprehensive survey on content-based image retrieval [31]. There are several commercial products for image similarity search as well. Incogna [59], Piximilar [58] and Riya [91] search similar images mostly by color and shape. Tineye [116] is a reverse image search engine that indexes about two billion images. It can detect partial duplicate images on the web from the user provided query image. Both Google and Live search engines have incorporated similarity search for a subset of the images they possess.

Many applications have relied on audio similarity search to find relevant songs

from a large music collection. Casey has written a survey on current research and the future direction of content-based music information retrieval [21]. In the commercial world, Shazam [99] allows users to search its database of 8 million songs. A user can use the smart phone to "listen" to a short clip of a song played on the radio. And then the Shazam app will query its database to figure out what the song is. Soundhound [107] lets a user sing or hum the song of inquiry in order to search its song database for a match. While both systems perform similarity search, they have different focuses: Shazam requires almost exact match of the song with allowance for noise and compression loss, while Soundhound requires a much less strict match of the original song, as exemplified by the user's hum.

Video similarity search systems started to appear in the 1990s with systems such as VideoQ [45], QBIC [42], and Zhang's integrated retrieval and browsing system [139]. In the last decade, with the explosive growth of home video, there has been an increased interest in content-based video search. Specifically, the yearly TRECVID [100] challenge, started in 2003, provides a large collection of videos and uniform scoring procedures to encourage researchers to compete in video retrieval tasks. Similarity search systems have been implemented with different focuses such as: whole video near duplicate detection and copy detection [134, 71, 131], video clip fingerprinting [72], concept based retrieval [105] and visual similarity search [137]. Geetha has also written a comprehensive survey [47] on "Content-based Video Retrieval".

Similarity search systems with large-scale high dimensional data also find applications in scientific data analysis. For genomic data, Eisen proposed hierarchical clustering [114] using average linkage to group genes with similar genome-wide expression data from DNA micro-array hybridization. Self Organizing Maps [115] and Mutual Information Relevance Networks [18] also define the similarity over the whole gene expression vector. But researchers seek to discover group of genes that co-express in different subsets of experiments. Recent work has proposed an approximation

method using a bi-clustering algorithm [26] to discover low variance sub-matrices of the complete data matrix.

1.3 Contributions

The goal of this dissertation is to study how to effectively combine multiple modalities to implement similarity search systems for large-scale datasets. Since this is a broad topic, we take a pragmatic approach by studying three individual systems each requiring different integration of multimodal features. This allows us to investigate several issues in multimodal data fusion. For each case, we attempt to answer the following questions:

- How to build a system to effectively combine results from multiple features?
- How to combine text features with non-text features effectively for similarity search?
- How to help a system designer model a growing system and predict the accuracy of the system using sampled data?

The first system, VFerret, allows users to search continuous archived personal video data. The goal is to allow users to search digital copies of personal memory and find valuable information in the future. The challenge is that users archive the personal video continuously and they don't have time to segment and annotate the video properly. We designed the VFerret system to allow users to search the video by similarity and time range. The results show that search accuracy could be improved to an average precision of 0.79 by combining audio and visual features, compared to 0.46 and 0.66 by independently using audio or visual features individually. By using a compact data structure (sketch), the system could easily hold the audio and visual features in memory for a year's worth of video on a personal laptop computer.

The second system, Image Spam Detection System, detects image spam in email. Due to the success of text spam detection systems, spammers moved to image spam which is much more difficult to detect due to the limitations of current computer vision techniques. We take a different approach that uses near duplicate detection to identify image spam. We rely on traditional anti-spam methods to catch a subset of spam images, and then use multiple image spam filters to detect all images that "look" like the spam images caught by traditional methods. Our prototype system achieved an 89% detection rate with a 0.001% false positive rate. The traditional computer vision based method had a similar detection rate but a much higher false positive rate of around 1%.

The third system, Product Navigation System, helps users find the desired products. Traditional product navigation systems use a text based search engine to locate labeled products. However, since product databases are growing fast and product tags are often ambiguous, searching for a specific product can be time consuming and cumbersome for users. New methods work by leveraging the images associated with the product to help the search. The challenge is to build a search system to combine text-based search and content-based image search to help customers quickly find the right products. We propose a search system that combines text search with content-based image search to improve navigation through the product hierarchy tree found in most e-commerce sites. We evaluated such a design with product data from a top e-commerce site and found that our design reduces the number of user clicks by 60% compared to current standard practice.

During the implementation of these systems, we encountered one common problem — the datasets are large and keep growing. It is often difficult for the system designers to properly size the resources for the system ahead of time. In order to address this problem, we designed a rank-based model for sizing sketches, compact data structures which approximate the original features. The model is designed to

help system designers understand how to choose sketch size properly with only a subset of data available, and to ultimately design more efficient large-scale multimodal systems.

Since multiple modalities are intrinsic in multimedia non-text data, we would like to investigate how to leverage them when designing similarity search systems. Although we have not found a general solution to using multimodal data in similarity search systems, this dissertation shows that it is possible to substantially improve search accuracy and efficiency by combining domain specific knowledge of multimodal data. We have studied the problem using three real systems, and they demonstrated much better accuracy compared to systems that did not leverage multimodal data. The following chapters will describe each similarity search system in detail.

Chapter 2

Video Search with Visual and Audio Features

2.1 Introduction

A challenge in building a digital memory system is the ability to search desired information quickly and conveniently. In 1945, Vannevar Bush described his vision of Memex [120]: "a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility.". Today, commodity disks and video cameras can easily be used to implement the first part of Bushs description of the Memex – continuously capture and achieve a persons life. The challenge is to design and implement a retrieval system that "may be consulted with exceeding speed and flexibility," realizing the second part of the Memex vision.

Much of the previous work on building retrieval systems for continuous archived data is based on attributes, annotations, or automatic classifications. These approaches have limitations in different dimensions. Attributes such as time and location are helpful for information retrieval, but they do not describe the content of

the archived information. Annotations of non-text data (such as audio, images, and video) can provide a text-based search engine with effective ways to retrieve information. However, generating annotations for a continuous-archived life log is a daunting and perhaps impossible task for most people. Automatic classifications try to generate annotations automatically. They tend to generate coarse-grained classes, whereas retrieving information in a life-long continuous archive require both coarse-grained and fine-grained content-based search capabilities.

Princetons CASS (Content-Aware Search Systems) project studies how to build a retrieval system to perform content-based search of a variety of data types. We have designed a system called VFerret which provides the ability to perform content-based similarity search on unlabeled continuous archived video data. We are interested in several research issues. First, we are interested in studying how to use both audio and visual features to effectively perform content-based similarity search on continuous archived video data. Second, we would like to design a system that requires minimal metadata to handle years of continuous archived data. Third, we would like the system to allow users to combine the content-based similarity search capability with annotation/attribute-based search in retrieval tasks.

This chapter describes the design and implementation of the VFerret system. The system segments the video data into clips and extracts both visual and audio features as metadata. The core component of the system is a content-based similarity search engine constructed using a toolkit called Ferret [79] that searches the metadata by a combination of filtering, indexing and ranking. The system also has a built-in attribute/annotation search engine that allows users to perform a combination of attribute/annotation-based and content-based search.

To experiment with the system, we used the DejaView Camwear model 200 as a capturing device to continuously record 6 weeks of a graduate students life. To evaluate the search quality of our system, we have manually labeled the data and

used a portion of the data as our training dataset. Our evaluation shows that the system can achieve an average precision of 0.46 by using visual features alone, 0.66 by audio features alone, and 0.79 by combining visual and audio. Our analysis shows that the current configuration of the system requires only 13.7Gbytes of storage for the metadata of 10 years of continuous archived data.

2.2 Previous work

Retrieving continuous archived data is an active field. Traditional methods use various kinds of attributes and annotations to aid retrieval. Mylifebits uses extensive attributes and annotations to create links between video, audio and all personal information together. Lifelog presents a system using GPS, body sensor and voice annotation to index the video.

There have also been various projects working on content-based video retrieval. Marvel [103], VideoQ [45], and most notably various projects participating TRECVID workshop [86] use visual and audio features and apply the content-based search technique to retrieve video clips. These projects mostly focus on commercial video clips, which pose different kind of challenges than personal continuous archived video.

Content-based audio retrieval research (e.g. Ellis and Lee [38]) has tended to focus more on automatic classification, clustering, and segmentation problems than on generic similarity search. Some work has been done towards defining similarity spaces for shorter-timescale sound effects and instrument tones (e.g. MARSYAS3D [97], Terasawa, Slaney, and Berger [113]). The problem of developing longer-timescale similarity metrics for music is also being actively studied (see e.g. Logan and Salomon [75], Vignoli and Pauws [123]).

Content-based image retrieval research has studied various visual features to find similar images. The initial QBIC [42] and many other content-based image search

system surveyed by Smeulders [102] studied quality of different image feature representations. More recently, region-based image retrieval like Blobworld [20] and local feature-based image retrieval like PCA-SIFT [67] demonstrated better retrieval performance. Since these methods are much more computationally intensive, we did not adopt them in our video search system.

Research on home video segmentation [133, 140] and organization [56] investigated techniques to organize home video which shares some similar characteristics with continuous archived video. Further investigation is needed to apply these methods to continuously archived video.

2.3 System Overview

When designing the system, we first examine a number of usage scenarios of continuous archived data to identify the needed functionalities. The following is a set of sample tasks a graduate student wants:

- Find the talk given by a professor during an industrial affiliate day.
- Recall the ideas proposed by a particular team member during a project group meeting.
- Find the clip where I met some baby Canadian geese along a trail.

For the first question, if one can find the specific date when the industrial affiliate day is from her calendar, it should be easy to find the clip quickly by browsing through the clips on that day.

The second and third tasks, however, will be more difficult and time-consuming with attribute/annotation-based search method if the video clips are not rigorously annotated. What we are proposing is to use content-based search to locate these

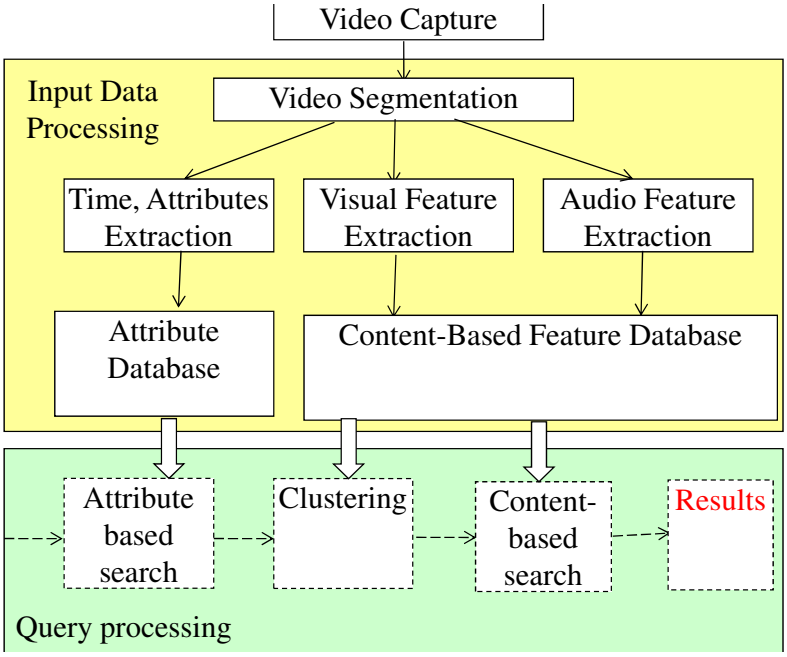


Figure 2.1: VFerret system architecture

clips: we can first find some clips that look or sound similar to our memory of the desired clips and then use content-based similarity search to find the clips of interest.

With these questions in mind, we built the video search system as shown in Figure 2.1. Once the continuously captured video is inserted into the system, it is first segmented into short video clips. We extract capture time and other possible attributes associated with the video clips and insert them into the attribute database. Meanwhile, in order to do content-based similarity search, we also extract the visual and audio features from the video clips so that we can index the clips based on their content.

At the query processing time, the user can combine the content-based search with attribute-based search to find the clips of interest. We believe these two processes are complementary: Attribute-based search can help to bootstrap a content-based search, while content-based search can search all the clips that have similar contents so that the clips that are not labeled can be found conveniently.

Our preliminary experience shows that a user typically starts with an attribute-based search with a time range; the resulting clips are then clustered based on visual and audio features. For each cluster, only one representative video clip is shown, so the user can quickly locate a video clip that is similar to the desired clip. Once a similar clip is found (this usually is much easier than finding the precise desired clip), the user can initiate the similarity search to find the clips of interest quickly. We will describe the system in more details in section 4.

2.4 Content-Based Similarity Search

We use content-based search as the main retrieval method to handle the unlabeled personal continuous archived video. The content-based search takes one video clip as a query clip, and returns a collection of video clips similar to the query clip. For example, using one meeting clip as a query object, one can find most other meeting clips without annotations. Our content-based video search system combines visual and audio features to determine the overall similarity of the video clips.

2.4.1 Video clip segmentation

The system first segments the continuous archived video into short video clips. This is a necessary step since the lengths of the original recordings vary and each recording can contain multiple activities.

We use a simple segmentation method to evenly split the video recordings into 5 minutes clips each. This is adequate for our search purpose since most of the video clips of our interests last more than 5 minutes. Although a better video segmentation tool would be more desirable, we leave this to future system improvements. We have tried several available commercial and research video scene detection and segmentation tools such as Handysaw [28], Microsoft movie maker, and the segmen-

tation tool from the authors [85]. These segmentation methods do not work well for continuous archived data because they depend on camera or lens movements that commercial videos tend to have for segmentation. We believe personal continuous archived videos are inherently different from commercial videos in terms of segmentations. Commercial videos have relatively clean shots and clear edited boundaries between scenes, whereas personal continuous archived videos tend to have unclean shots and no editing.

2.4.2 Visual feature extraction

For each segmented video clip, the system extracts a set of visual feature vectors to represent the clip. These features are used in the similarity search to determine whether two video clips look similar or not.

To extract the visual feature, we evenly sample 20 individual images from each video clip. For the video clips segmented into 5 minutes each, we extract one frame every 15 seconds. For each frame, we first convert them from RGB into HSV color space since the HSV color space distance is better for measuring human perceptual similarity. To compare images for similarity, the system uses the approach proposed by Stricker [109] which uses 3 central moments of the color distribution. As shown in Figure 2.2, the 3 moments are mean, standard deviation and skewness, which describe the average, variance and the degree of asymmetry of the color distribution. Ma [82] has shown that the color moments performs only slightly worse than the much higher-dimensional color histogram.

In our system, we take 3 color moments of each channel of HSV space. This gives us a compact 9 dimension feature vector for each image. With the training dataset, we further normalize the feature vector with their mean and standard deviation. Finally, we use L1 distance to calculate the distance between feature vectors.

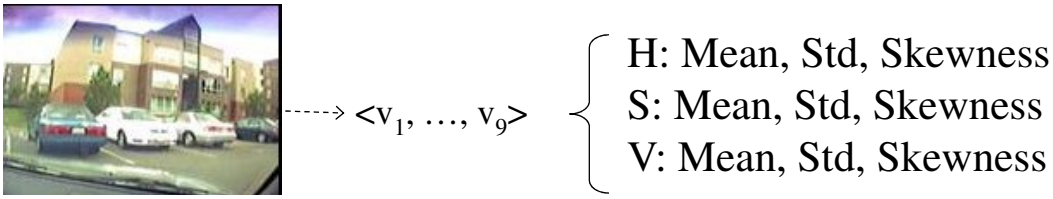


Figure 2.2: Visual feature extraction

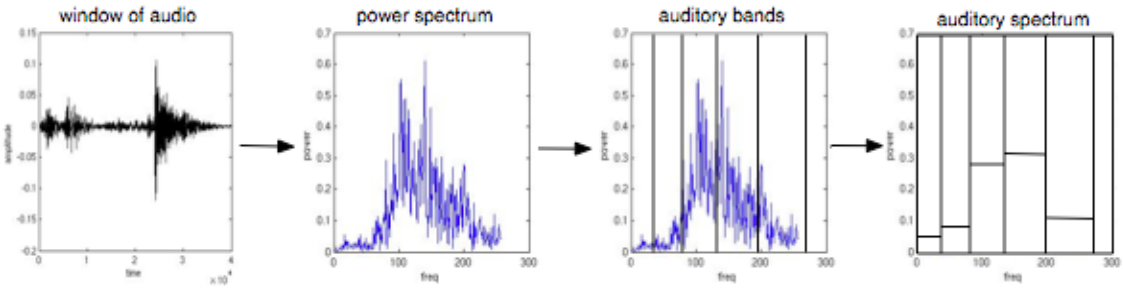


Figure 2.3: Audio feature extraction

2.4.3 Audio feature extraction

To extract audio features, the system evenly split the audio channel of each 5-minute video clip into 20 individual 15-second segments.

For each 15-second audio segment, the system uses 154 audio features patterned after those used by Ellis and Lee [37] to describe audio segments. Figure 2.3 shows the extraction process. We begin by extracting several sets of short-time features describing 10 ms windows calculated every 5 ms over the entire segment. Then, to condense this information into a compact descriptor of the entire segment, we take the means and standard deviations of these short-time features, normalizing the standard deviations by their respective means.

The first set of 25 short-time features measures the energy in each of 25 Bark-scale frequency bands of the window. The Bark scale divides the frequency spectrum into bands that increase in width with frequency in a way that models the bandwidth of our auditory system, as shown in Figure2.3. To measure the energy in one of these bands for a given window, we take the short-time Fourier transform (STFT) of the

window and sum the energy in all frequency bins that fall within that band.

The next set of 25 short-time features further describe each Bark-scale band by treating the energy spectra within those bands as probability distributions and calculating their entropies. If all of the energy in a band is concentrated within one bin, its entropy will be very low, whereas if the energy is more evenly distributed the entropy will be high.

For our last set of 25 short-time features, we again treat the energy spectrum within each band as a probability distribution and calculate the Kullback-Leibler divergence for each band between subsequent windows. This provides us with information about how much the shape of the spectrum within each band is changing from window to window.

Finally, we calculate the entropy and Kullback-Leibler divergence as above for the entire short-time Bark-scale energy spectrum, yielding another 2 short-time features.

Taking the means and normalized standard deviations of each of these 25 + 25 + 25 + 2 features gives us our 154 long-time audio features:

- 50: Mean, std of energy in each of 25 Bark-scale band
- 50: Mean, std of entropy in each of 25 Bark-scale band
- 50: Mean, std of Kullback-Leibler divergence in each of 25 Bark-scale band
- 4: Mean, std of entropy and Kullback-Leibler divergence for the entire energy spectrum

L1 distance is used to calculate the distance between feature vectors.

2.4.4 Combined feature vector

For each 15-second video segment, we combine the visual feature vector extracted from the sample image and the audio feature vector extracted from the corresponding

audio segment to form a single feature vector. The proper weight assigned to visual and audio features are derived from the training data set as described in section 5.1. We use L1 distance to calculate the distance between the combined feature vectors.

2.4.5 Similarity search

Given a query video clip, the goal of the similarity search is to find all the clips that are similar to the query video. In our system, we represent each video clip as a set of visual and audio features. So given the query video clip X, we would like the system to find all video clips Y in the collection such that the distance d(X,Y) is within a small range (also referred to as k nearest neighbor problem). The similarity search system will return a ranked list of video clips where the clip with smallest distance to the query clip ranks first.

Since each video clip is represented by a set of combined feature vector rather than a single combined feature vector, we need to find a proper distance between set of feature vectors. In our implementation, we use the one-to-one best match method to find the overall minimal distance between two sets of feature vectors.

As shown in Figure 2.4, query clip X is sampled into individual images and audio clips. A set of feature vectors $< X_i >$ are extracted from the clip, one from each image and audio segment. Same applied to all the other clips in the collection. For a particular candidate Y, the distance d(X,Y) is defined as the best one-to-one match such that the sum of all distances between the underlying feature vectors is minimized:

$$d(X,Y) = min\{\sum_{j=1}^{n} d(X_i, Y_{f(i)})\} \tag{2.1}$$

Where $f(i)$ is a function provide any permutation of $[1, ..., n]$, $d(X_i, Y_j)$ is the distance between the corresponding feature vectors.

During the similarity search, all the video clips in the collection are compared with

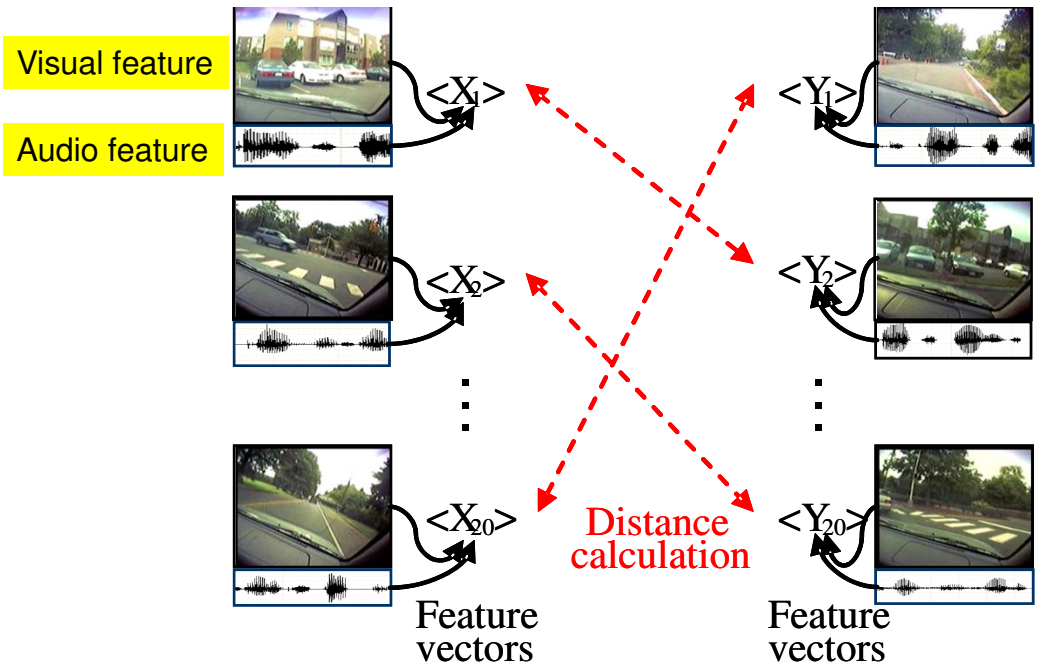


Figure 2.4: Illustration of distance computation

the query video clip. They will be ranked according to their distance to the query video clip. To speed up the similarity search, our system uses sketches to represent feature vectors, and perform filtering and indexing to speed up the search process. A technical paper on the Ferret toolkit [79] provides more detailed information.

2.5 VFerret: Content Search for Continuous Captured Video

2.5.1 Video capture system

We adopt the commercial wearable camera [1] as shown in Figure 2.5. The DejaView Camwear model 200 has a separate camera lens that can be attached to hat or eyeglass, and a recording device that can record up to 3-4 hours of video with a single charge of battery. It records 320*240 mpeg4 video clips with sound to the secure digital flash memory card. One hour of video will take about 0.5 GB of storage space.

Figure 2.5: DejaView Camwear model 200 system

One of the authors carried the camwear, and recorded on average of 1 hour of video every day from May to June.

2.5.2 Video search system

The video storage and search system is built using the Ferret toolkit. Our system leverages the existing Ferret infrastructure by configuring it with video segmentation, visual and audio feature extraction components.

To fully utilize the content-based similarity search, it is important to start a similarity search with a relevant query video clip. We have implemented attribute-based search methods to help bootstrap the content-based search quickly. These methods can reduce the number of video clips users need to browse through, but they still present a challenge when many video clips remain to be checked. The role of content-based similarity search is to bridge the gap between the results returned from attribute-based search and the final results.

We will use an example to illustrate the search process. Consider the example that someone wants to show a friend the clips where she saw several baby Canadian

geese with their family on her way home.

Timeline-based search step

The timeline-based search is the most natural method to search the personal continuous archived data. Since the time stamp comes for free and people naturally anchor events with time, most systems for personal archive have this capability. In our experience, the timeline based search is effective when the event has a distinctive date (e.g.: Christmas) or is associated with some other context (e.g.: email, event saved on calendar) that is searchable via other means. The Mylifebits system [49] and the Lifelog system [111] leverage the context information and demonstrate the effectiveness of using timeline and context to retrieve contents.

On the other hand, for old events or relatively insignificant events, it is difficult to recall the exact time it occurred. In such cases, one must use a relatively wide time range, yielding many candidate video clips. A time range can be used to reduce the search range in the first step of our video search system. For our example, the user recalls that the encounter happened early this summer. So the user can limit the search from May 1st to Jun 1st, which will reduce the number of clips in the next step.

Clustering step

After the timeline filtering, the candidate set may still be too large for a user to browse through quickly. Our system uses a k-means clustering algorithm [3] to cluster the filtered candidates into a small set of clusters. A representative video clip is found from each cluster so that user can quickly browse the full collections.

The k-means algorithm uses the same visual and audio features as the similarity search system. The only difference is that we use the average of the 20 feature vectors, rather than using all of them. This reduces the size of the overall feature vector and

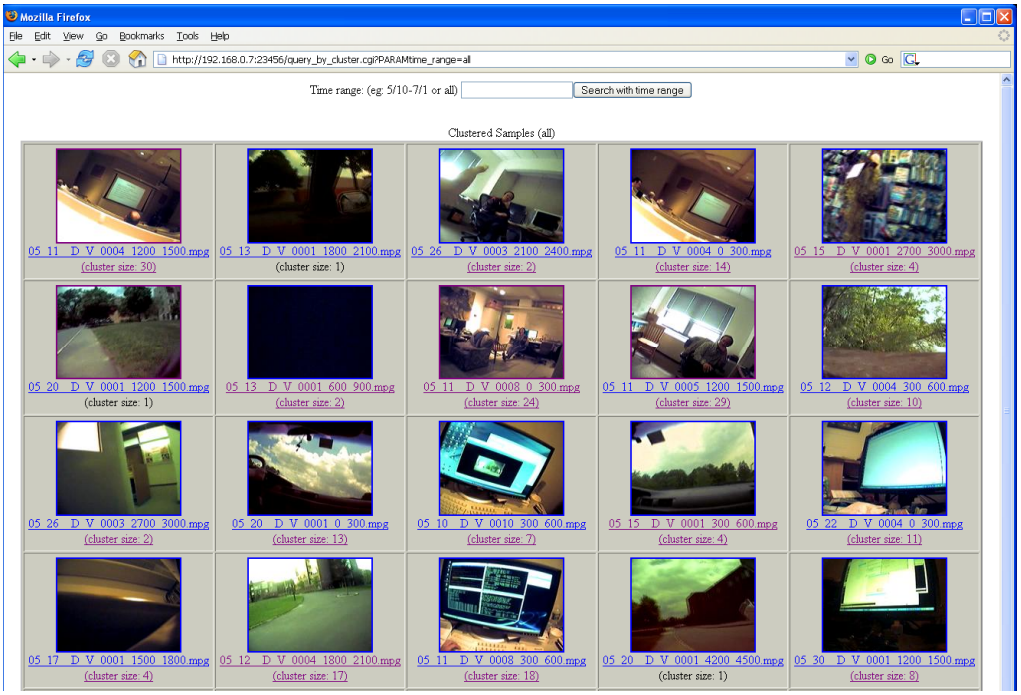


Figure 2.6: 20 video clusters presented after timeline and clustering steps

greatly speeds up the clustering speed to make it interactive. This design decision is based on the observation that clustering is for users to choose candidates to perform content-based search queries instead of final results. So long as the user can identify one video clip that is similar to the desired clip, she will be able to start the similarity search with that clip.

For our example, the video clip should be an outdoor scene and is on a trail with lots of green trees. The user will look for a cluster with outdoor scenes. Figure 2.6 shows an example of clusters presented to the user.

Content-based similarity search step

The last step of the search is the content-based similarity search. Once user has a query clip, she can initiate the similarity search and iteratively refine the search to find the desired result. She can either use a new clip in the search result as the new query, or use multiple similar clips to start a new search. This process will provide

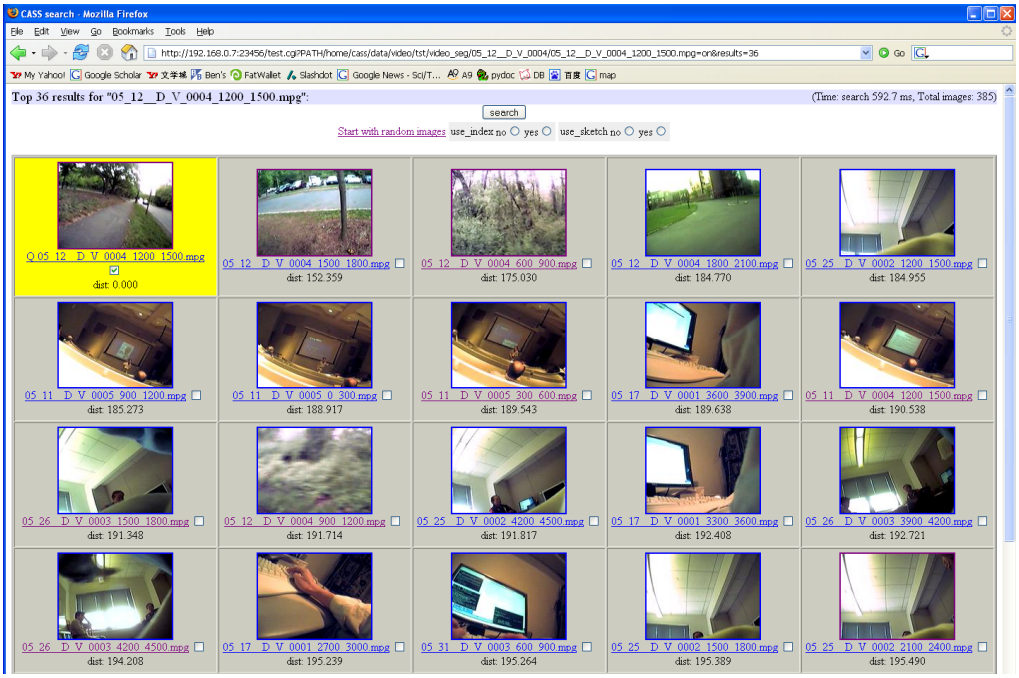


Figure 2.7: Results after the content-based similarity search step

higher quality results iteratively and help the user quickly pinpoint the desired clips without browsing through the entire candidate set.

For our example, the user would get a collection of similar trail video clips and find the clips of interest. Figure 2.7 shows a snapshot of such search result.

2.6 Evaluation

We have done an initial evaluation of the system to answer two questions:

- How well does the content-based similarity search produce high-quality results?
- What is the systems resource overhead for the content-based search of continuous archived video?

One of the authors recorded 6 weeks of his personal life using the Camwear gear. The video clips involve activities such as work, drive, walk, meeting, shopping, etc.

Activity label	Training set (Number of clips)	Test set (Number of clips)
walking outside	5	2
meeting	9	7
shopping	6	5
driving	20	7
seminar	10	10
reading	4	4

Table 2.1: Similarity sets

The system segments the data into a total of 385 video clips, as described above.

2.6.1 Benchmark

We separate the video clips into two sets of about the same size: one for training and one for testing. For each set, a similarity benchmark is manually defined. The training set is used for training the system while the test set is used here to report the benchmark result.

For the benchmark, we defined the similarity sets by manually reviewing the video clips and grouping video clips together according to activity types to form similarity sets. For example, one similarity set consists of several clips recorded while walking on an outdoor trail while another similarity set consists of recordings made while driving in a car. Within a similarity set, all the clips are believed to similar to each other, thus no rank is given within the similarity set. Note that, the benchmark clips are only a subset of all the video clips and all video clips are used in the similarity search test.

We come up with 6 similarity sets for each case, labeled as in Table 2.1:

2.6.2 Evaluation metric

We have chosen to use average precision to measure the effectiveness of our similarity search. Given a query q with k similar clips where query q is excluded from the similarity set, let $Rank_i$ be the rank of the i_{th} retrieved relevant clip ($1 <= i <= k$)

Feature vectors	Average Precision
Visual	0.46
Audio	0.66
Visual + Audio	0.79

Table 2.2: System Results

returned by the system, then average precision is defined as follows:

$$average\_precision = \frac{1}{k}\sum_{i=1}^{k}\frac{i}{Rank_i}$$

Suppose similarity set is q1, q2, q3 and the query is q1. If the search results returned by the system are r1, q2, q3, r4, the average precision is $1/2 * (1/2 + 2/3) = 0.583$. This measure provides a single-valued measure, simplifying comparison of the effectiveness of different systems.

2.6.3 Results

We compared the average precision result of our search system using visual features alone, audio features alone, visual and audio features together as shown in Table 2.2:

Our results on the benchmark suggest that the audio features are contributing more to our search performance than the visual features. This is an interesting result, and we believe that it comes from the fact that in our benchmark, audio can capture more environmental features than visual. Although the camwear lens provides 60 degree field of view, the captured video still varies a lot in the same environment as head moves around. Meanwhile, audio captures relatively stable features independent of the head position in the same environment. This gives audio more power to distinguish different environments which are associated to the activities in our benchmark.

Although audio feature works well in classifying activities, we still rely mostly on visual part to present the search interface.. The current interface allows the user

to see a tile (8x8) of thumbnails created from the video clip to quickly grasp the visual content of the clip. For audio part, we do not have such capability of fast-forwarding or quick sampling of the full clip. As a result, the visual feature still plays an important role in users search process.

2.6.4 System overhead

The similarity search system only needs to store extra feature vectors in addition to the video clips for similarity search capability. Even if the user continuously record the video 24 hours a day and 7 days a week, it will only need about 1.37 GBytes extra storage space to store the feature vectors for one year worth of video.

For the search speed, the current system can return the similarity set within 600ms for a collection of 385 clips. No indexing or filtering is used for the current search since linear scan is fast enough for 385 clips. In order to search tens of years of continuous archived video, we believe with timeline based search to reduce the search range and Ferrets filtering and indexing capability to speed up search, the query should still be answered in the order of seconds.

2.7 Summary

This chapter presents the design and implementation of VFerret, a system that provides content-based similarity search for unlabeled continuous archived video data. Our initial evaluation with a simple benchmark shows that the system can perform high-quality content-based similarity search. While using visual and audio features individually can achieve 0.46 and 0.66 average precisions respectively, combining both can achieve average precision of 0.79.

The metadata overhead of the system is small. The current implementation uses about 1.4GB of metadata for content-based similarity search for one-year worth of

continuous archived video data. This implies that it is already practical to implement content-based similarity search in a current computing device.

Chapter 3

Spam Detection with Multiple Image Features

3.1 Introduction

Spam message volumes have doubled over the past year and now account for about 80% of the total messages on the Internet. A major reason for the increased prevalence of spam is the recent emergence of image spam (*i.e.* spam embedded in images). Image spam volumes nearly quadrupled in 2006, increasing from 10% to 35% of the overall volume of spam; worse, the volume of image spam continues to rise[27, 110]. The situation has significantly frustrated end-users as many image spam messages are able to defeat the commonly deployed anti-spam systems. In order to reduce the impact of spam, it is crucial to understand how to effectively and efficiently filter out image spam messages.

Spammers have recently begun developing image-based spam methods to circumvent current anti-spam technologies since existing anti-spam methods have proved quite successful at filtering text-based spam email messages. Early image-based spam simply embedded advertising text in images that linked to HTML formatted email so

that its content could be automatically displayed to end-users while being shielded from text-based spam filters. As spam filters started using simple methods such as comparing the hashes of image data and performing optical character recognition (OCR) on images, spammers have quickly adapted their techniques. To combat computer vision techniques such as OCR, spammers have begun applying CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) techniques. These techniques distort the original image or add colorful or noisy backgrounds so that only humans can identify the intended message [19]. Once spammers have applied an image creation algorithm to make a message difficult to detect with computer vision algorithms, they apply further randomization to construct a batch of images for delivery. The additional randomization defeats hash-based detection mechanisms. The result is that current image spam methods present serious challenges for anti-spam systems.

Although some research has been done to distinguish spam images from non-spam images by using computer vision techniques including filtering noisy images for recognizing embedded text or monitoring color saturations in an image[6], such methods tend to have high false positive rates, labeling ham (non-spam) as spam. This is because computer vision techniques have not been able to defeat CAPTCHA. Furthermore, it is difficult to predict what spam images will look like as they are constantly evolving to evade detection. In addition, sophisticated computer vision techniques often require substantial CPU resources, making them less practical in high-volume environments.

We believe that an effective image spam detection system should satisfy several requirements. First, it should be *accurate*, detecting most image spams while maintaining a low false positive rate. Second, it should be *efficient*, parsing incoming emails with images at modern WAN speeds. Third, it should be *extensible*, allowing new image spam filtering methods to be added to deal with quickly evolving image

spam techniques.

This chapter proposes an image spam detection system to satisfy the requirements above. The basic idea is to use traditional anti-spam methods to detect some image-based spam messages and then use fast near-duplicate detection filters to detect the variations of known spam images. The system we propose is based on two observations. The first is that traditional spam detection methods such as honeypots, message header analysis or human reporting mechanisms can detect some image spam messages. The second is that image spam messages are typically sent in large batches where the messages in each batch are visually similar, although the variations can be sophisticated. For example, spammers often design a template image and apply various randomized alterations or noise to the template before sending it out to each end user. Figure 3.3 and 3.4 show some spam image samples. We believe that this is because spammers still want to deliver clear information to end users and they want to use efficient methods to generate millions of unique spam images while not obscuring the template image too much.

Rather than studying the image itself to determine whether the particular image is a spam image or not, our system adopts an alternative approach. We use very efficient near-duplicate detection techniques to find spam images that are variations of other spam images caught by traditional anti-spam methods. Thus our system is complementary to the existing anti-spam system to help detect image spams missed by the traditional system.

We have designed and implemented a prototype of the proposed image spam detection system. The system supports the use of multiple image spam filters, allows users to plug in new filters and to specify different aggregation methods among the filters including AND (all filters agree), OR (one of the filters decides) and VOTE (certain number of filters agree). We have implemented three image spam filters using different near-duplicate detection techniques in our prototype system. Our

experiments with a suite of image spam benchmark and 10 million non-spam images show that using VOTE method can effectively detect variations of most kinds of image spam messages while maintaining the false positive rate to less than 0.001%.

3.2 Previous work

Nowadays, spam filters are widely deployed and various anti-spam techniques have been developed. At the network layer, systems such as Mail Avenger [83] track source, destination, network path, and software version information for analysis by spam filters. Many anti-spam systems also use a combination of whitelists, blacklists, and so-called greylists that force legitimate clients to re-send messages since spammers often do not bother doing so [73]. Other common techniques include block lists distributed via DNS that identify addresses assigned to dialup users or known open relays and challenge-response systems that automatically build whitelists. Most systems such as Mail Avenger, Spam Assassin, and SpamGuru [83, 108, 98] use multiple techniques, including multiple classifiers, to identify spam.

Filters for text-based spam, including plain text and HTML e-mail, have employed a variety of statistical techniques, particularly Bayesian inference [95, 54]; these statistical filters appear to classify text-based e-mail well. Another popular approach is the use of so-called "fuzzy signatures" such as those employed by Vipul's Razor [124] and the Distributed Checksum Clearinghouse [32]. Fuzzy signatures are designed in such a way that the signature for substantially different messages are unlikely to collide but that the signature for very similar, although not necessarily identical, messages will collide with high probability. Systems such as DCC allow users at different sites to share fuzzy signatures for reported spam.

Although image-based spam has been around for some time, recent reports and anecdotal evidence suggest that there has been significant growth both in the volume

of image-based spam and in the percentage of spam that uses image attachments to convey its message [89, 110, 61]. Unlike earlier image-based spam, current image spam is randomized to avoid signature based anti-spam techniques. Since the majority of spam is now delivered through botnets [90], spammers have the bandwidth and computational resources necessary to customize individual spam images extensively.

Although traditional spam filters that rely on analysis of sender, message header and various other information can detect some of the image spams without looking into the image itself, other image spams still pass through the spam filter since spammers try hard to make everything except the spam image itself look innocent. Some recent research studied the image classification using computer vision techniques. One approach[6] is largely based on the extraction of text regions in the images of interest and SVM. This method can achieve about 85% detection rate with about 15% false positive rate. Wu *et al.* [132] have identified a number of useful visual features including banner images, computer generated graphics and embedded-text, and use SVM to train the classifier. Their approach can achieve about 81% detection rate with about 1% false positive rate. Although the results are encouraging, we believe that the misclassification rate of non-spam images needs to be dramatically lower in order to make image spam detection practical.

Our image spam detection system utilizes content-based image similarity search techniques. Much work has been done in this area [36, 94, 101, 121]. Recently, researchers have used content-based image search techniques for near-duplicate image detection [68, 138]. The image features they use are usually complex and slow to compute and compare. Given the large number of image spams that are received every day, our challenge is to achieve high effectiveness and high efficiency at the same time.

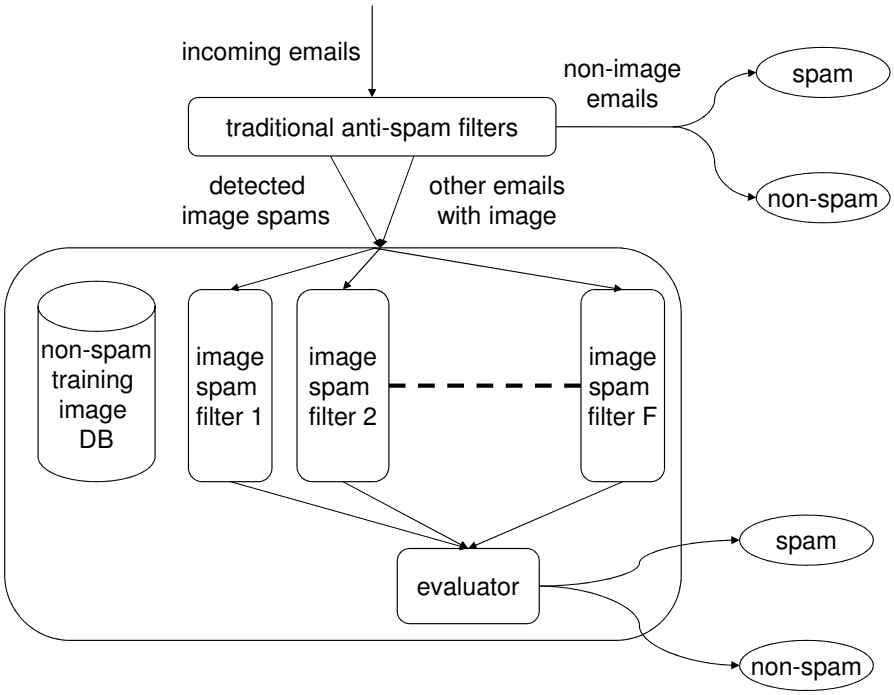


Figure 3.1: Image spam detection system architecture.

3.3 Main Idea

Our image spam detection system takes advantage of the nature of spam messages: they have to be sent in large quantity, and the machine generated spam images within the same batch will look similar to each other. Since spam messages are sent in large quantity, some of them can be detected using traditional spam detection methods such as honeypots (dummy email accounts set up to attract spams) or sender analysis, as well as those reported by end users. Once we have identified some of the spam images via traditional spam detection methods, we can then detect their near-duplicates as spams even if those emails evade traditional spam filters.

Figure3.1 shows the architecture of our image spam detection system: the traditional spam filters not based on image content analysis; a set of image spam filters that detect near-duplicate images; and an evaluator to aggregate the results from

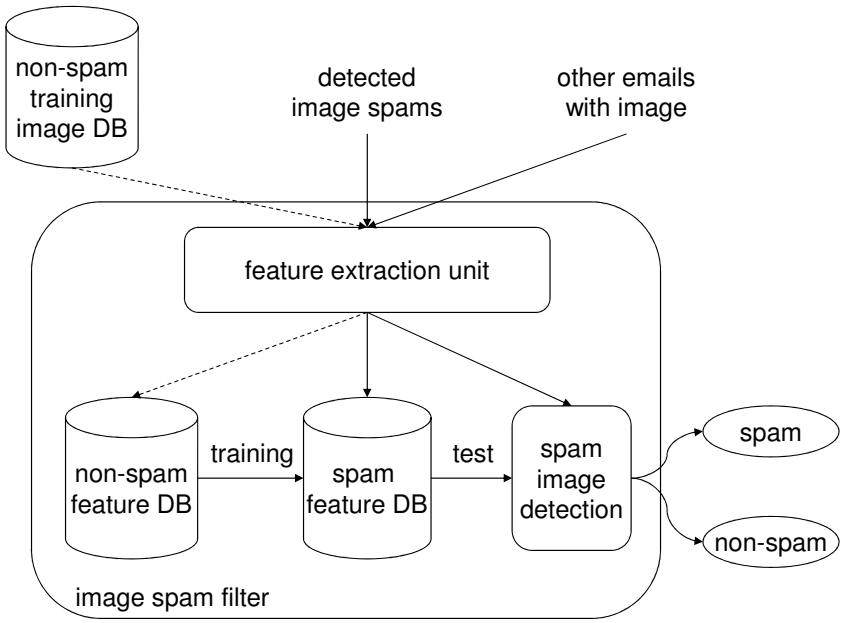


Figure 3.2: An image spam filter.

multiple image spam filters. Each image spam filter is made up by its feature extraction unit, non-spam feature DB, spam feature DB and spam image detection unit as shown in Figure 3.2.

An important issue in designing a near-duplicate detection system is to figure out how "near" when comparing two images with each other. In other words, we need to figure out a distance threshold in the filter's feature vector space of the images. If a threshold is too small, the filter tend to achieve low false positive rates, but low detection rates. Traditionally, threshold will be derived by training with a spam image dataset. We took a different approach, deriving thresholds by training with a non-spam image dataset. The idea is to learn what the "non-spam" images look like and use the information to bound the threshold to achieve high detection rates and low false positive rates. The intuition behind of this approach is that the look of non-spam images are relatively stable over time, whereas image spam methods

are constantly evolving and new spams may have different looks. We collect a large collection of known non-spam images and save it in the non-spam training image DB. Each deployed image spam filter (or a new image spam filter plug-in) will extract features from all the non-spam images offline and store them in the non-spam feature DB for future use.

When deployed in real time, the image spam detection system will work together with the traditional anti-spam filters. All incoming emails will go through the traditional anti-spam filters first. The emails without any embedded images will be handled by traditional filter alone. The emails with embedded images will be filtered and labeled first by the traditional spam filter (such as analyzing their headers) and then passed to our image spam detection system.

All embedded images will be processed by all image spam filters. The feature extraction unit of each individual image spam filter will extract a feature vector from the image. If the image is labeled as spam by the traditional anti-spam filter, the feature vector will be inserted into the spam feature DB. During the insertion, we will calculate its distance from all the feature vectors stored in the non-spam feature DB and set the smallest distance as the detection threshold associated with this particular spam image. This step will essentially create a high dimensional sphere in the feature space such that any other feature vector that falls into the sphere will be considered as a spam.

An image spam filter then uses the spheres defined by the thresholds in the spam feature DB to detect near-duplicates of the known spams. Conceptually, the image spam filter will compare the feature vector of an image (not labeled by the traditional spam filter) with all the feature vectors in the spam feature DB. If the feature vector falls in any of the "sphere" of a known image spam, the email associated with this image will be considered as a spam by this specific image spam filter.

We allow multiple image spam filters to work together to increase the coverage

of spam categories, improve the detection rate for each category while maintaining low false positive rate. Our idea is to use an evaluator to aggregate the results from multiple filters. We currently consider three methods:

- **"AND"**: an image is classified as spam if and only if all filters decide it is spam. This method will lead to relatively low detection rates and very low false positive rates.
- **"OR"**: an image is classified as spam if any filter decides it is spam. This method will lead to high detection rates and relatively high false positive rates.
- **"VOTE"**: an image is classified as spam if a specified number of filters decide it is spam. This method will provide balanced detection rates and false positive rates between AND and OR.

By supporting multiple aggregation methods, our system provides users with more flexibility.

3.4 Anti-Spam System: Image Spam Detection System

To evaluate our idea, we have implemented a prototype image spam detection system. This section describes the implementation details of our system components.

We use Mail Avenger [83], a customizable SMTP server, together with SpamAssassin mail filter [108] as the traditional spam filter in our system. Each incoming message was annotated with the output available from Mail Avenger. Mail Avenger was configured to deliver mail unconditionally but to log the TCP fingerprint and results of a traceroute back to the mail relay as well as the results of DNSBL lookups for about twenty different popular black lists. Once messages to the mail server were spooled, they were run through the SpamAssassin mail filter. The real-time feedback

from Mail Avenger and SpamAssassin were used to determine how existing tools would classify incoming spam *at delivery time*.

We store 100,000 legitimate images in our non-spam image DB. They are sampled from two online photo sharing sites photo.net and pbase.com and from a COREL stock photo collection. The images are stored in the database so that new image spam filter can be trained when introduced into the system.

Image spam filters are designed to detect variations of image spams using different image near-duplicate detection techniques. Our system supports multiple image spam filters and allow users to plug in new filters for emerging image spam techniques. Figure 3.2 shows the components of an image spam filter.

We have leveraged Ferret toolkit [79] to construct image spam filters, manage spam feature DB, and perform near-duplicate detections. It is convenient to use since it uses advanced indexing techniques to perform high-speed similarity searches. To construct an image spam filter, all one needs is to plug in a feature extraction unit and the definition of its distance function.

The feature extraction unit converts an image to a feature vector representation for near-duplicate detection purpose. If two images are near-duplicates, their feature vectors would be very close to each other in the feature vector space. The following properties are highly desirable for the feature extraction unit:

- *Efficient* : The unit should be able to process incoming images efficiently in order to match the throughput of targeted mail servers.

- *Effective* : Spammers typically add random "noises" to each spam image. For effective detection, the unit should produce features that are relatively insensitive to those added noises.

- *Distinctive* : To minimize false positive rate, the unit should generate features that can distinguish spam images from non-spam images.

We have constructed three filters (see Section 6).

The system has a spam feature DB for each filter. The DB stores all feature vectors extracted from all known spam images labeled by the traditional anti-spam filter, and an associated threshold value for each feature vector. The threshold value is the smallest distance between the feature vector of the spam image and the feature vectors of all the 100,000 non-spam images. If a new known spam image is within the threshold value distance away from an old known spam image, it will be treated as the spam from the same batch and no new entry will be inserted into the spam feature DB to save space. Older known spam feature could be retired if it had not been able to detect spams for a while.

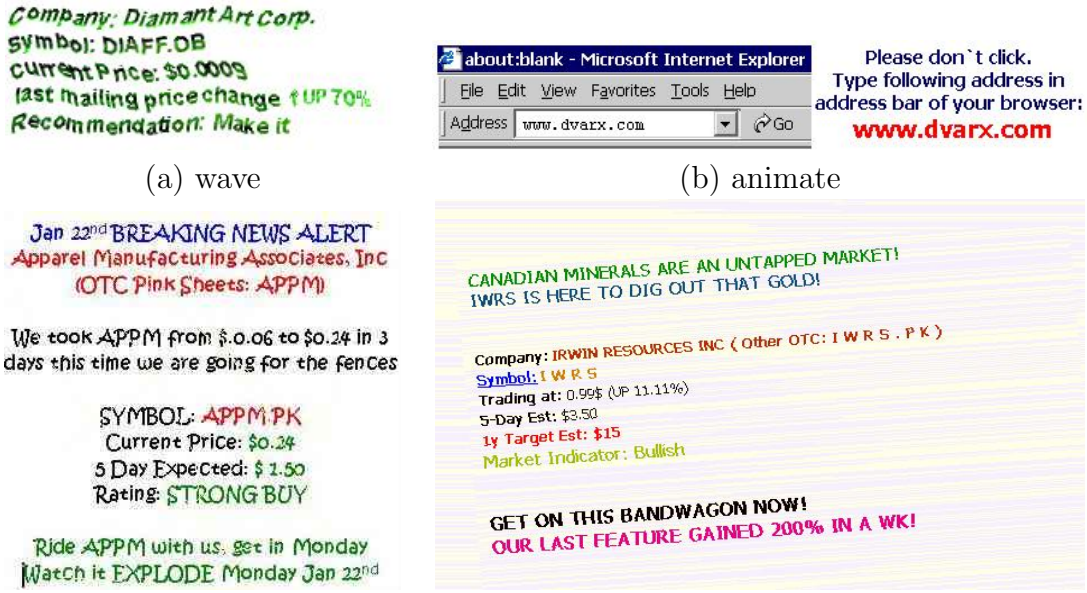
We have implemented three aggregation methods AND, OR, and VOTE in the evaluator. See previous section for their aggregation functionalities.

3.4.1 Known Image Spam Techniques

Spam images are typically generated in two steps. The first step is *template construction*, where a spam image template is constructed with the intended content for end user. The main goal is to use different methods such as CAPTCHA to defeat computer vision (such as OCR-based) anti-spam techniques. The second step is *randomization*, where a large number of spam images can be generated from the template image using various randomization techniques, in order to defeat signature-based anti-spam techniques. This section describes the template construction and randomization methods used in the image spam datasets we have collected.

Construction methods: among the spam images we have collected, we have identified four template construction methods:

- *wave* : See Figure 3.3 (a). This method uses wavy text to make it more difficult for OCR recognition.



(a) wave (b) animate

(c) deform (d) rotate

Figure 3.3: Examples of spam construction techniques.

- *animate* : See Figure 3.3 (b), the URL in the web browser is animated. By using animations in the GIF format, it is harder to detect the real spam content.

- *deform* : See Figure 3.3 (c). This method uses deformed text (such as irregular handwritten fonts, different font colors) in order to defeat OCR.

- *rotate* : See Figure 3.3 (d). This method rotates the text to a certain angle such that it is not horizontal and more difficult for OCR. Depending on the number of different angles can be used, this technique can also be used as a randomization technique (see below).

Randomization methods: The goal is to add random noises to a template spam image in order to generate a large number of spam images to defeat traditional signature-based anti-spam techniques. From the spam images we have collected, we have observed 17 randomization techniques:

- *shift* : template image is shifted horizontally or vertically on the canvas. See Figure 3.4 (1), (7).

Figure 3.4: Example spams belongs to different categories of spamming techniques.

- *crop* : template image is cropped differently (sometimes sacrificing part of content). See Figure 3.4 (2), (3) and (13).
- *size* : slight variations of the size (height and width) of the template. For example, this can be achieved by writing the same template content on canvas of different sizes or resizing an image. See Figure 3.4 (4), (10).
- *dots* : adds random dots (or speckles). See Figure 3.4 (5), (11), and (13).
- *bar* : adds a randomized bar (of pixels with similar colors) to the top, middle, or end of a template image. See the blue bars in Figure 3.4 (4).
- *frame* : adds a frame (of randomized pixels and different thickness) to the template image. See the frames in Figure 3.4 (5).
- *fonttype* : uses different font types for the text. See Figure 3.4 (5), (7).
- *fontsize* : uses different font sizes for the text. See Figure 3.4 (11), (12).
- *fontcolor* : uses different font colors for the text. see Figure 3.4 (8). Further randomization is achieved by using different colors within each individual letter. See Figure 3.4 (7).
- *line* : uses randomized lines in the background. See Figure 3.4 (11).
- *linecolor* : uses randomized lines of different colors in the background. See Figure 3.4 (6), (7).
- *shape* : uses randomized shapes (such as polygons or ellipse) in the background. See the pink shapes in Figure 3.4 (3).
- *rotate* : rotates the text to a random angle. See Figure 3.4 (6).

- *bits* : uses a few randomized bits either in the metadata or at pixel level, resulting in different image files but no noticeable different images. See Figure 3.4 (14).

- *content* : uses different wording (but of similar theme) for each line in a multi-line message to achieve a combined high level of randomness. See Figure 3.4 (8).

- *fuzzy* : uses fuzzy text and lines. See the fuzzy text and discontinued lines in Figure 3.4 (10).

- *url* : uses different URLs (pointing to the same products). See Figure 3.4 (9), notice the only difference is the URL.

Figure 3.4 shows the samples of spam images, some of which use combinations of the methods above[1].

3.4.2 Image Spam Filters

Since spammers use different randomization methods to introduce noises to spam images, a single feature might not be able to effectively detect all variations. In our system, we have experimented with 3 filters based on color histogram, wavelet, and orientation histogram.

Color Histogram Filter

The color histogram is a simple feature and can be calculated efficiently by one simple pass of the whole image. We have used 64-dimensional color histogram based in the RGB color space. Values in each of the three color channels (R,G,B) are divided into 4 bins of equal size, resulting in $4 \times 4 \times 4 = 64$ bins in total. For each bin,

[1]We did not collect enough of the "slice-and-dice" image spams (where the spammer randomly slice the original spam image into several small pieces) mentioned in other report to form a batch.

the amount of color pixels that falls into that particular bin is counted. Finally it is normalized so that the sum equals to one. We use L_1 distance to calculate the distance between two color histogram features. For images represented by D-dimensional real-valued feature vectors, the $L1$ distance of the pair of points $X = (X_1, \ldots, X_D)$ and $Y = (Y_1, \ldots, Y_D)$ has the form:

$$d(X, Y) = \sum_{i=1}^{D} |X_i - Y_i|$$

We adopt color histogram in our system for its simplicity and efficiency.

The color histogram is effective against randomly added noises and simple translational shift of the images. For the spam randomization techniques described in section 3.4.1, the color histogram is designed to handle shift, size, dots, bar, frame, fonttype, fontsize, line, rotate, bits, content, fuzzy, url.

Haar Wavelet Filter

2-D Haar wavelet transformation is popular in image analysis and can be calculated efficiently in $O(n)$ time. We convert the color image into a 256×256 grayscale image and apply 2-D Haar wavelet transformation on it. We then take the first 4×4 wavelet coefficients at low resolution end of the matrix. This essentially provides the low resolution information about the original image. $L1$ distance measure is used to calculate the feature distance.

The Harr wavelet feature is mainly targeted for these randomization techniques: size, dots, bar, frame, fonttype, fontsize, line, shape, bits, content, fuzzy, url.

Orientation Histogram Feature

Orientation histogram feature provides the histogram of orientation of edges in the image. It was shown to be effective in hand gesture recognition in [44] and can be

calculated by one simple pass of the image. We start by calculating the orientation of each pixel, then bin the orientation of each pixel into 36 groups, each of which is 10 degrees. After that, we use a 1 4 6 4 1 filter to blur the orientation histogram. The final histogram is normalized and $L1$ distance is used to calculate the feature distance.

Orientation histogram is designed to work better with these randomization techniques: shift, crop, size, fontsize, fontcolor, linecolor, bits, url.

3.5 Evaluation

We would like to answer the following questions:

- How effective is each image spam filter?
- How well do multiple image spam filters work together?
- What are the performance implications of these filters?
- How efficiently can we propagate spam image signatures for distributed spam detection?

To answer these questions, we have conducted experiments with our prototype system using a collection of spam images and non-spam images.

3.5.1 Evaluation Datasets

We use two different kinds of images in our evaluations: spam images and legitimate ("ham" or non-spam) images. Since many new kinds of image spams have emerged recently, the image spam techniques used a year ago are substantially different from the current ones. We have decided to create an image spam dataset using image spams collected during recent three months (Dec. 2006 to Feb. 2007) instead of using an old public spam corpus.

Our spam images are collected from seven different email accounts. These are personal email accounts including accounts from two popular online webmail service providers, one IT company account, and three education accounts. All images in the spam emails, including user identified ones, are collected. Duplicate spam images are removed by using SHA-1 hash; some malformed images are also removed. All the remaining images are manually classified into batches based on their content. We also extract the time stamp at the receiver for each image to verify the batches or further split the batches whenever necessary. We have removed batches that contain only a single image for our benchmark. The resulting spam image dataset contains 1071 images in 178 different batches. The min, max, average and standard deviation of the batch sizes are 2, 50, 6.02, 6.39 respectively. The full spam image benchmark is available online [88].

In order to evaluate false positive rates, we have constructed a non-spam image dataset. Since there are few publicly available non-spam email repositories (especially for emails containing images) due to privacy concerns, we have used samples of photos downloaded from popular photo sharing web sites as our non-spam images. We use two sets of non-spam images, one for training and one for testing:

- Training non-spam dataset: 100,000 image randomly selected from over 600,000 images downloaded from PBase and Photonet, and the COREL stock photo collection.

- Testing non-spam dataset: 10 million images downloaded randomly from the Flickr web site.

We believe they are a good representation of non-spam images sent via email.

Category	fpos%	det%
dots	0	100.0
shift, fonttype, dots	0.00010	100.0
shift, bar	0.00012	100.0
bits	0.00016	100.0
shift, fonttype, dots, frame	0.00023	100.0
shift, dots, url	0.00044	100.0
fontsize, dots, line	0.00046	100.0
shift, dots, line	0.00077	100.0
shift, fuzzy	0.00118	100.0
size, bar	0.00126	100.0
size	0.00126	100.0
shift, url	0.00128	100.0
size, dots	0.00161	100.0
shift	0.02401	99.1
shift, dots, fontsize	0.00259	97.7
shift, dots	0.00617	97.4
size, fuzzy	0.00381	96.7
crop, dots, shape	0.00012	100.0
shift, linecolor, rotate	0.00014	100.0
shift, linecolor, fontcolor	0.00039	100.0
crop	0.00167	100.0
shift, linecolor	0.00390	92.8
shift, linecolor, fontcolor,fttype	0.03452	42.8
shift, content, fontcolor	0.00009	0.0
crop, dots	0.00027	0.0
overall	0.08655	84.7

(a) Color Histogram

Category	fpos%	det%
bits	0.00002	100.0
size, bar	0.00003	100.0
fontsize, dots, line	0.00003	100.0
dots	0.00004	100.0
size	0.00012	100.0
size, fuzzy	0.00032	100.0
size, dots	0.00088	100.0
shift, dots, url	0	100.0
shift, fonttype, dots, frame	0.00001	100.0
shift, fuzzy	0.00003	100.0
crop	0.00004	100.0
crop, dots	0.00005	100.0
shift, fonttype, dots	0.00009	100.0
crop, dots, shape	0.00053	100.0
shift	0.01138	92.3
shift, dots, fontsize	0.00013	88.6
shift, dots	0.00061	85.9
shift, linecolor, rotate	0.00001	75.0
shift, linecolor, fontcolor,fttype	0.00936	69.2
shift, url	0.00004	62.5
shift, content, fontcolor	0.00001	60.0
shift, linecolor	0.00012	56.5
shift, dots, line	0	50.0
shift, linecolor, fontcolor	0.00006	25.0
shift, bar	0.00002	0.0
overall	0.02393	82.3

(b) Haar Wavelet

Category	fpos%	det%
shift, linecolor, fontcolor	0.00029	100.0
bits	0.00052	100.0
crop	0.00061	100.0
shift, url	0.00065	100.0
shift, linecolor	0.00179	94.2
shift	0.02089	88.2
size	0.00052	83.3
shift, linecolor, rotate	0.00028	100.0
fontsize, dots, lint	0.00031	100.0
shift, dots, url	0.00060	100.0
shift, bar	0.00065	100.0
dots	0.00073	100.0
shift, fonttype, dots	0.00077	100.0
size, fuzzy	0.00106	100.0
shift, dots, fontsize	0.00081	89.8
shift, dots	0.00286	89.1
shift, content, fontcolor	0.00016	80.0
crop, dots, shape	0.00007	75.0
shift, linecolor, fontcolor,fttype	0.02616	62.2
shift, dots, line	0.00011	50.0
shift, fuzzy	0.00075	50.0
crop, dots	0.00007	33.3
size, bar	0.00082	33.3
size, dots	0.00103	33.3
shift, fonttype, dots, frame	0.00109	28.6
overall	0.06360	81.6

(c) Orientation Histogram

Table 3.1: Results using different image spam filters. The categories shown in bold are the "targeted" group for each filter.

3.5.2 Individual Image Spam Filter Results

We begin by evaluating the effectiveness of each individual image spam filter in isolation. To evaluate the effectiveness of a filter, we present the system with a single marked spam image, the remaining unmarked spam images, and 10 million non-spam images and see if our filter can detect the unmarked spam images in the same batch. Table 3.1 shows the false positive and detection rates for each spam category using different image spam filters. For each filter, the first group (shown in bold) contains the categories that the particular filter is designed to handle while the second group shows the remaining categories. The last row shows the overall false positive and detection rate for all the categories.

The results show that all three filters did well in "targeted" categories in terms of detection rates and false positive rates. The color histogram filter was able to

achieve perfect detection rates for 13 out of 17 targeted categories, and more than 96.7% for the remaining 4 categories. The false positive rates of all categories except one (shift) are below 0.006%. The wavelet filter achieves perfect detection rates for all targeted categories while keeping the false positive rates below 0.0009%. The orientation histogram filter achieves perfect detection rates for 4 out of 7 targeted categories, while keeping the false positive rates below 0.0007%. The detection rates for the remaining 3 categories are 94.2%, 88.2% and 83.3% with false positive rates 0.00179%, 0.02089% and 0.00052% respectively.

The filters achieve good detection rates in more than half of the "un-targeted" categories. The main reason is that although a filter is not designed specifically to handle these categories, spammers tend to be conservative in randomizing images since they must preserve the readability of the spam messages. This makes makes some of the randomization techniques less effective. All individual filters achieve low false positive rates.

3.5.3 Combined Image Spam Detection Results

To study the effect of aggregating multiple spam filters, we have experimented with three simple aggregation methods: "AND", "OR", and "VOTE". Table 3.2 shows the results using all three methods to aggregate results from multiple spam filters. The results are presented in two groups: the first group is the union of all "targeted" categories from three spam filters, the second group shows the remainder of the categories. The last row shows the overall false positive and detection rates for all the categories.

We can see that the "AND" method consistently achieves a false positive rate of zero, but is effective in only about half of all spam image categories. Because individual filters are focusing on different features of the images, when all filters agree collectively, we expect a minimum false positive rate. The low false positive rate can

	Evaluator	AND		OR		VOTE	
Category	# spams	fpos%	det%	fpos%	det%	fpos%	det%
size,bar	23	0	33.3	0.00211	100.0	0	100.0
crop	15	0	100.0	0.00232	100.0	0	100.0
shift,dots,url	12	0	100.0	0.00104	100.0	0	100.0
size,dots	12	0	33.3	0.00352	100.0	0	100.0
dots	9	0	100.0	0.00077	100.0	0	100.0
size	9	0	83.3	0.00190	100.0	0	100.0
shift,linecolor,fontcolor	6	0	25.0	0.00074	100.0	0	100.0
shift,fonttype,dots	4	0	100.0	0.00096	100.0	0	100.0
shift,fuzzy	3	0	50.0	0.00196	100.0	0	100.0
bits	2	0	100.0	0.00070	100.0	0	100.0
fontsize,dots,line	2	0	100.0	0.00080	100.0	0	100.0
shift,url	26	0	62.5	0.00196	100.0	0.00001	100.0
shift,fonttype, dots,frame	8	0	28.6	0.00132	100.0	0.00001	100.0
shift,bar	2	0	0.0	0.00078	100.0	0.00001	100.0
size,fuzzy	36	0	96.7	0.00517	100.0	0.00002	100.0
shift,dots,fontsize	100	0	78.4	0.00353	100.0	0	97.7
shift,dots	185	0	75.0	0.00963	100.0	0.00001	97.4
shift	276	0	83.3	0.05610	100.0	0.00018	96.4
shift,linecolor	76	0	55.1	0.00579	97.1	0.00002	91.3
shift,dots,line	3	0	50.0	0.00088	100.0	0	50.0
shift,linecolor,rotate	7	0	75.0	0.00043	100.0	0	100.0
crop,dots,shape	5	0	75.0	0.00072	100.0	0	100.0
shift,linecolor, fontcolor,fonttype	240	0	26.9	0.06958	83.6	0.00046	63.7
shift,content,fontcolor	6	0	0.0	0.00026	100.0	0	40.0
crop,dots	4	0	0.0	0.00039	100.0	0	33.3
overall	1071	0	63.6	0.17336	96.1	0.00072	88.9

Table 3.2: Results using different evaluators to aggregate results from multiple image spam filters.

be useful for certain use cases.

The "OR" method delivers the best spam detection rates at the cost of higher false positive rates. It detects most of the spam images, including most of "un-targeted" ones. For some use cases, a false positive rate of 0.17% is considered tolerable, but other use cases require lower false positive rates.

The "VOTE" method provides a compromise between false positive and detection rates; it holds the false positive rates below 0.0002% for all targeted categories,

while achieving good detection rates. The only category that it didn't do well is shift.dots.line, which exhibits a 50% detection rate. For the un-targeted categories, VOTE keeps false positive rates below 0.0005% and has good detection rates (63.7%, 100%, 43%, 100%, and 33%, respectively). We can understand effect of VOTE better if we study one particular category closely, say "shift". We get false positive rates of about 0.024%, 0.011%, 0.021% and detection rates of about 99.1%, 92.3%, 88.2% from three filters. After VOTE, we can achieve a false positive rate of 0.00018% and a detection rate of 96.4%.

Our results show that multiple filters can work better than an individual filter. For example, the VOTE method can deliver a better overall detection rate than each individual filter, while reducing the overall false positive rate by almost two orders of magnitude compared to each individual filter. This supports our design goal of making the system extensible.

3.5.4 Image Spam Filter Speed

	feature extraction time (ms)	training time (ms)	detection time (ms)
Color histogram	20.9	19.8	2.0
Haar wavelet	5.4	9.4	1.1
Orientation histogram	14.5	14.4	1.5

Table 3.3: Image Spam Filter Speed.

In order to understand the performance implications of the image spam filters, we have measured the processing time for the main components of our system on a P4 3GHz test machine. Table 3.3 shows the processing time for each image filter: image feature extraction time (assuming the image is rendered into memory ahead of time), training time where a new "known" spam image is inserted into the system and its threshold value is determined by comparing with 100,000 known non-spam images, and detection time where an incoming image's feature is compared with the

features in the spam image feature database (we assume there are 10,000 spams in the database). Note that the training time is taken only for new kind of "known" spam image, thus it does not occur every time a known spam image is inserted. On average, a new image will take less than 50ms to be processed through all filters.

3.5.5 Image Spam Signature Size

Since the proposed image spam filters use feature vectors for near-duplicate detection, it is possible to distribute new feature vectors to end mail server systems over the Internet. Each participating email server can send its newly detected spam image "signatures" to the central server which aggregate the spam image signatures and periodically broadcast them back to email servers.

To see how practical our method is for supporting collaboration between peers, we have calculated the network overhead for exchanging information about new spams. In our approach, only the image feature and the associated threshold value generated by the spam filter need to be exchanged over the network. The three spam filters require $(64 + 36 + 16 + 3) \times 4 = 476$ bytes per known image spam.

3.6 Summary

In this chapter, we present an image spam detection system. By examining the content of new images contained in incoming emails and detecting images that are near-duplicates of known spam images, our system can effectively detect image spams while maintaining a low false positive rate. Rather than using computationally expensive algorithms to detect new types of image spams designed to thwart conventional computer vision algorithms, our system uses efficient algorithms to target randomization methods used to generate large number of unique but visually similar image spams from template images. Our system is designed to be integrated with existing

anti-spam technologies to boost the detection rate of image spams. Our prototype system has demonstrated high detection rates in most spam categories while achieving a less than 0.001% false positive rate using the "VOTE" aggregation method.

Chapter 4

Product Navigation with Image and Text Features

4.1 Introduction

A major challenge for an e-commerce system is enabling users to search for desired products quickly. A popular approach taken by most existing systems is to arrange products in a manually maintained tree of categories, and to index product tags and descriptions with a text search engine. This allows a user to locate the desired products by combining tree navigation and text search. However, as the number of products grows larger, level-by-level navigation of the category tree becomes a frustrating task. Further, it is non-trivial for users to identify appropriate keywords to initiate a search for a product they have in mind such as a particular style of dress or furniture.

With the popularity of smart phones with cameras, an emerging approach is to allow users to search for products using images. However, computer vision techniques for recognizing objects are effective only when the number of categories is relatively small [33], and so far successful applications of image search are limited to a few special

types of products. For example, "Google Goggles" [51] only supports searching book covers and wine labels which are essentially 2D images.

In this chapter, we present an approach to combine text search with image search to implement a mechanism for customers to search for products of interest using both text and images. The main idea is to leverage the existing manually labeled product hierarchy tree and combine its navigation mechanism with a content-based image search engine. The system uses the content-based image search results to help the user predict the right branches of the tree. The user can correct the predictions at any level. This will iteratively narrow the scope of the search and eventually zero in on the products of interest. This method guarantees that the user makes predictable progress toward the goal of finding the relevant product and improves the accuracy of content-based image search by narrowing down the scope of the search.

Our contributions in this chapter are threefold. First, this is one of the first papers to study various designs in combining text search and image search for large e-commerce sites. Second, we propose a new system to use image search to assist the user in navigating the product tree and locate the desired product quickly. Third, we evaluate our system using a dataset from a large e-commerce site that contains more than 30 million products and show that our system reduces the number of user clicks needed to locate a particular product by about 60% over the traditional approach.

4.2 Previous work

Our system makes use of content-based image similarity search technique. For general content-based image retrieval, readers are referred to Datta et al. [31] for a comprehensive survey. For the image near duplicate detection task (a special case of image similarity where the objective is to detect whether or not a picture is derived from the original picture with minor graphic modification such as cropping, blurring or

increasing contrast), the detection accuracy is good as demonstrated by commercial web sites such as Tineye [116], and various research prototypes [68, 127].

On the other hand, object categorization still poses a major challenge due to variations in objects, background and viewpoints. For the well-studied dataset Caltech 101 [64] with 101 categories, the classification accuracy had been improved to around 70-80%. However the object classification accuracy drops quickly as the number of categories increases. A recent evaluation with Imagenet [33] showed that the classification accuracy is below 10% for 10,000 categories. An e-commerce site can easily have tens of thousands of different product categories, which is beyond the capability of today's vision technique.

There have been few published results on image search of products (probably due to potential commercial applications). Jing and Baluja [62] suggested using the pagerank algorithm on product image search results generated by text search to improve user experience. Goswami et al. [52] studied the impact of product images on user clicks for online shopping. Our proposed research system is different from previous research since it suggests a completely new method to navigate the product hierarchy tree with the help of image search.

4.3 System Architecture

The goal of a product search system is to allow users to quickly locate products of interest. We accomplish this goal by combining text search and image search. We first discuss the design choice we made to properly combine the text based search with image search.

The traditional content-based image search system typically uses an image as the query and searches the whole dataset looking for similar images. In fact, most of the current commercial content-based image search systems such as "Google Goggles"

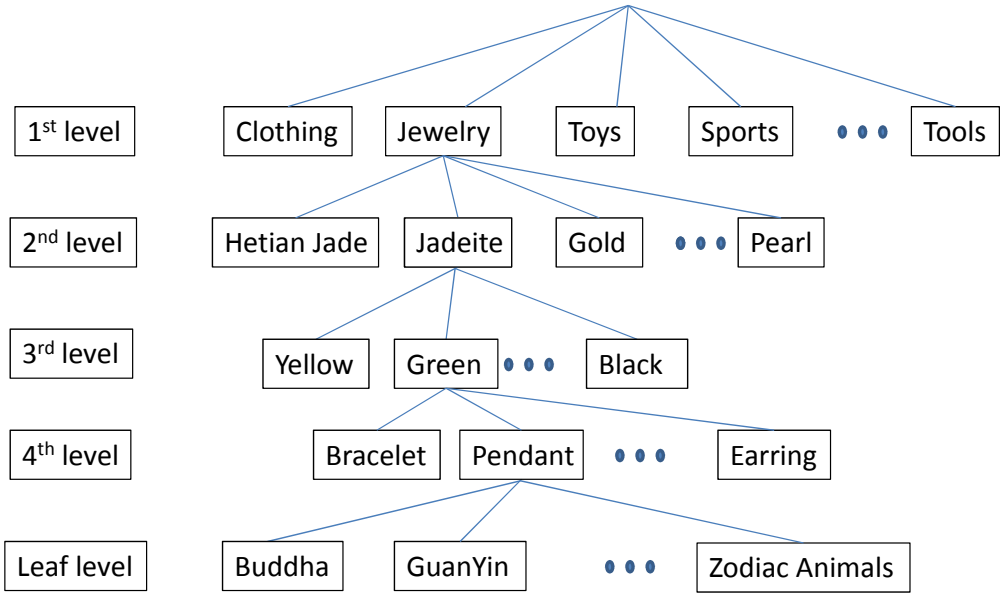


Figure 4.1: Product Hierarchical Tree

and Amazon Mobile App [4] adopt such a method. When potential customers cannot find the products they want from the similar product images returned by the system, they give up the image search system and switch back to the traditional product keyword based system. We consider this the naive way of combining the text based search system with content-based image search system. The user basically tries to use the image search system first, and uses the traditional text based system as a fallback method. This is not an ideal solution since presumably, the reason the user tried the image search in the first place was that the text based search is not very effective for the kind of query the user is making.

When we designed our system, we took advantage of the following observation: For most e-commerce sites, almost all of the products are already manually categorized into a hierarchical product tree such as the one shown in Figure 4.1. We designed our system to use the image search to assist navigation through this type of product tree. After the user submits the query image, the system generates a list of candidate product category labels for the user to choose from. The user can help choose the

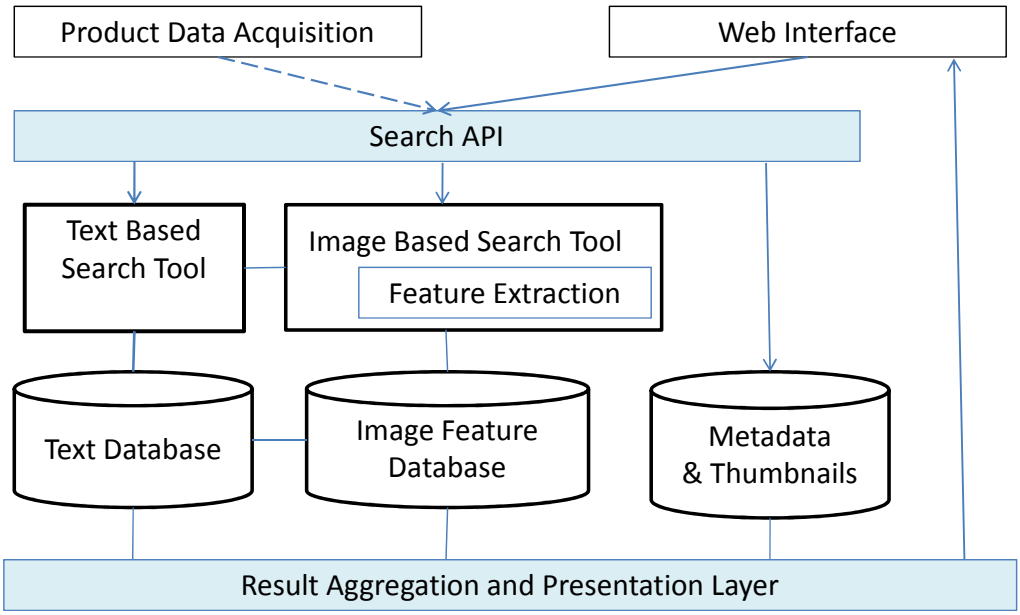


Figure 4.2: System Architecture

correct product category to narrow down the search scope iteratively until reaching the correct product. There are several benefits of such an approach: first, the category labels are usually the most useful tags for the products, since they are manually designed by the site operator to minimize the effort required to navigate through the products. Second, the search scope narrows as the user gets deeper into the product tree, and consequently, the accuracy of the image search result is progressively improved. Third, the user is guaranteed to get to the desired product category in a limited number of steps.

Figure 4.2 is a functional block diagram that shows the relationship between the main components:

- *Product Data Acquisition* collects the product data from an e-commerce site and injects them into our system.
- *Web Interface* provides a web interface for the end user.
- *Search API* provides a thin layer API for inserting product data and searching.

- *Text Based Search Tool* implements a traditional searchable index on product text tags. It can provide a subset of the full dataset such that the *Image Based Search Tool* can continue to refine the results.

- *Text Database* is the data storage of product text information used by *Text Based Search Tool*.

- *Image Based Search Tool* implements a content-based image search engine which enables the search for similar images given a query image. It achieves such a capability by extracting image features in such a way that similar images tend to have similar image features. Thus performing nearest neighbor search in the image feature space will find similar images to the query image.

- *Image Feature Database* is the data storage of image feature vectors used by *Image Based Search Tool*.

- *Metadata and Thumbnails* contains the metadata and thumbnails of the products used to construct the search results page returned to the user.

- *Result Aggregation and Presentation Layer* aggregates the results from the Search Tools and presents them to the end user.

4.4 Image Assisted Product Navigation System

In this section, we discuss the details of our system implementation.

4.4.1 System Workflow

We illustrate the system work flow in Figure 4.3. First, all the existing products are indexed into the search engine databases. We extract product category label information and store them in the text database. At the same time we also extract

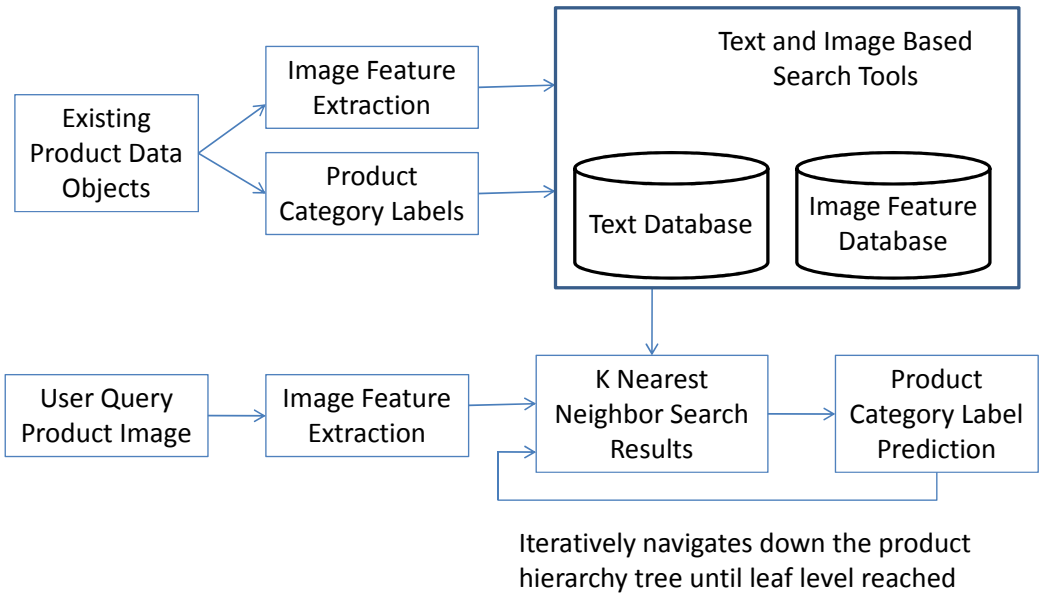


Figure 4.3: Detailed System Flow Chart

image features from each product image associated with the product, and store them in the image feature database. When a user submits an image of a new product to start the search process, we will first extract image features from the query image, and use these features to initiate a K-Nearest-Neighbor similarity search to find the most visually similar product images in the image feature database. Since we already know the product category label of these similar images, we can use that information to predict the most likely first level product category label. We will create a candidate list of product category labels with the most likely product category label on the top of the list.

During the user interaction step, the list of the candidate product category labels is presented to the user for confirmation. Since there are typically a limited number of category labels at each level of the product category tree, even if our search system makes a mistake in category label prediction, the user can still correct it by choosing the right category label to continue the search. Once the first level category label is selected, the system will do another round of K-Nearest-Neighbor image similarity

search in the reduced subset of product images where only the images under the correct first level category will be considered. This significantly reduces the amount of images to be searched and the process will continue until the leaf level category label is correctly identified. The user will finally be presented with a list of products in the leaf level category with the most similar product at the top.

4.4.2 Image Search Subsystem

For the image search subsystem, we have adopted three kinds of global image features, which were selected for their simplicity and speed to satisfy the real time performance requirements for our system:

- *Color Histogram* : We use a simple 64-dimensional color histogram based on the RGB color space. Values in each of the three color channels are divided into 4 bins of equal size, resulting in 64 bins in total. The amount of pixels that falls into each bin is counted to form a histogram.

- *Edge Histogram* : This is the standard MPEG-7 edge histogram descriptor. Each image is divided into 4×4 subimages, and for each subimage, the detected edges are divided into 5 bins: vertical, horizontal, 45 degree, 135 degree diagonal and non-directional edges. This gives us a total of $16 \times 5 = 80$ bins. We used the Lire [77] library to extract this image feature.

- *Average Intensity Histogram*: The image is first turned into gray scale and divided into 8×8 subimages; then the average intensity of each subimage is recorded. This actually gives us a tiny 8×8 "thumbnail" view of the original image.

In our image search system, we normalize each histogram and concatenate them to form a single 208 dimensional feature vector. We use L_1 distance to calculate

the distance between the features. For images represented by D-dimensional feature vectors, the $L1$ distance of the pair of points $X = (X_1, \ldots, X_D)$ and $Y = (Y_1, \ldots, Y_D)$ has the form:

$$d(X, Y) = \sum_{i=1}^{D} |X_i - Y_i|$$

4.4.3 User Interface

An important component of our search system is the user interface to help a user navigate the product tree with the assistance of image search. It should allow a user to navigate the product tree intuitively and correct prediction error easily. We adopt a multiple column user interface similar to the Mac OS X finder's column view. Figure 4.4 shows an illustration of our user interface. The columns show the product labels from the product tree at different levels. The rows are grouped in two regions: the first region lists the most likely candidate labels predicted by image search system, and the second region lists all the remaining candidate labels.

When the user submits a product image as the query image, our system provides a list of candidate first level category labels shown in the first column. To reduce the number of user clicks needed to navigate to the leaf level category, we also automatically fill the second column with candidate second level category labels assuming the top choice of the first level category labels is correct. This process happens recursively until it hits the leaf level category and the most similar product image from the top leaf level category is shown as the candidate.

In the example shown in Figure 4.4, the user submits a photo of a pendant as the query, and our system fills the multiple column interface with predicted category labels. It predicts the category labels correctly at the first level, but makes a mistake at the second level. As a result, all the subsequent category predictions are also incorrect. Because the users know what they really want, they correct the category

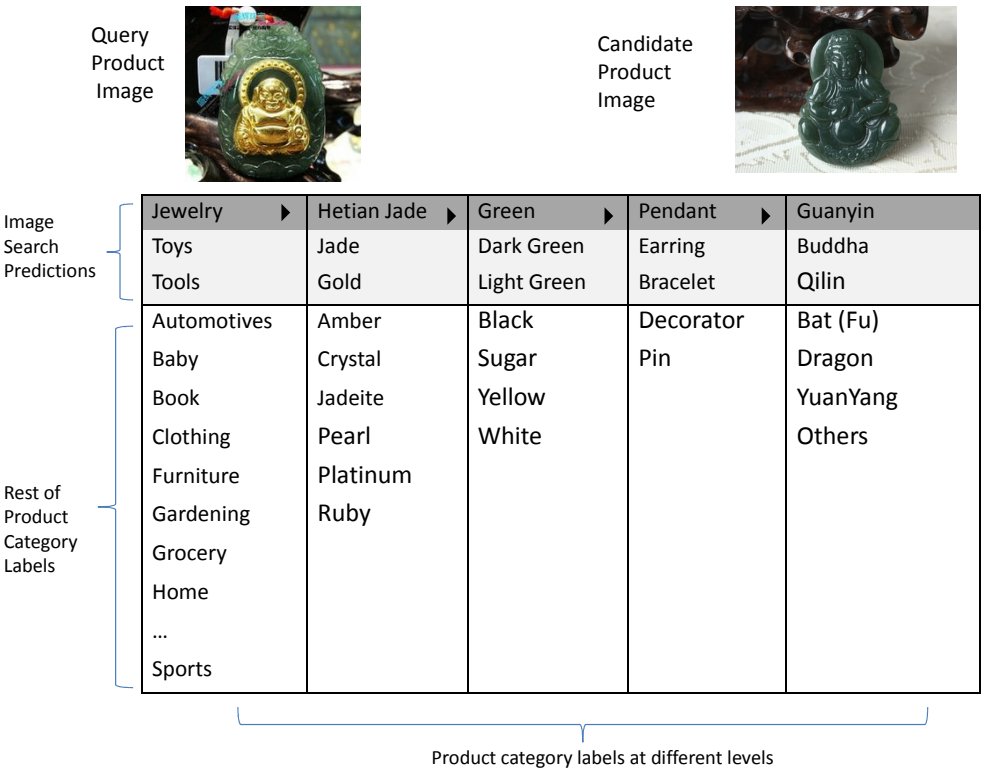


Figure 4.4: User Interface

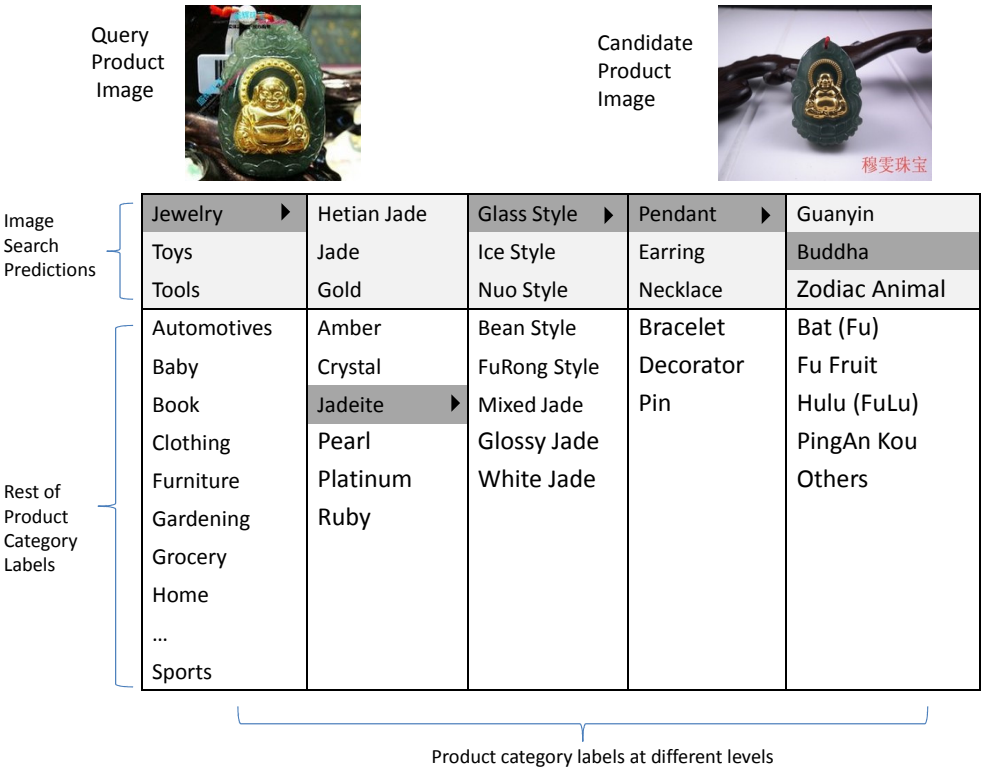


Figure 4.5: User Interface (After User's Correction)

label at the second level. As shown in Figure 4.5, the user's correction triggers a refresh of subsequent columns and these columns are filled with new predictions based on the user's correction at the second level. After the user's correction at the second level, the third and fourth levels are correctly predicted. Although the fifth column is not predicted correctly, one more user click fixes it and leads to the correct leaf level category. As shown in this example, though the underlying image prediction does not always work, the user only needs to click two times to correct the system and reach the desired leaf category level.

4.5 Evaluation

We would like to answer the following key question by evaluating our system with real data from a top e-commerce web site:

- How much improvement can our search system achieve compared with a traditional system?

4.5.1 Evaluation Dataset

We have taken a snapshot of the biggest Chinese e-commerce web site: Taobao [112] during May 2010. The Taobao website is similar to Ebay, but is more heavily regulated with detailed manual product categorization. It allows any vendor to sell almost any products on its website. As a result, the leaf level category tends to have the same products since lots of different small vendors are selling the same product while competing for price and service. We remove the virtual product categories such as prepaid game points card, service, insurance products from the dataset to focus on product search for real physical products.

Our Taobao dataset contains about 30 million product images after removing duplicate images. We only take one product image for each product. The dataset has

Product tree level	1st	2nd	3rd	4th	5th	6th	7th
Average number of categories	69	15.3	9.9	9.1	7.8	8.8	7.8

Table 4.1: Average Number of Categories at Different Levels in Product Hierarchy Tree

69 first level categories and about 350,000 leaf level categories. The average number of categories from the same parent category at different levels in the product hierarchy tree is shown in Table 4.1. The average number of products at the leaf level is around 172 and the average depth to reach the leaf level category from the top is 5.1.

4.5.2 Experimental setup

We leverage the manually labeled product category labels for each product to create our benchmark. From our Taobao dataset, we randomly sample 600 product images, remove them from the dataset and use them as query images to initiate the search. We use the existing category labels for the 600 products as the ground truth, and compare them against the prediction made by our search system.

We use the average number of user clicks needed to navigate to the leaf level category to evaluate the effect of using different systems on user experience. Unlike Amazon which aggregates different vendors of the same product into the same product page, Taobao organizes all these different vendors in the same leaf level product group. Thus in most cases, the correct leaf level category will give the user the right product in our Taobao dataset. We use the average number of user clicks to compare the performance of the following systems:

Text Based System

The text based system is representative of how most existing e-commerce websites work. The products are arranged in a hierarchical product tree, and the user is greeted with the home page with the first level category labels to start the search.

The user will navigate through the product tree structure one level at a time until they hit the leaf level category. We assume that each web page will be large enough to display all category labels at that category level. We count one user click for each category level traversed.

Since we do not have detailed log information about tag based search results, we can not evaluate the effectiveness of a system that uses tag keyword search to locate the products. It is possible that for some products with a distinctive name such as "Canon 7D" camera, the keyword based search will work best. However for several other products, the user simply does not know the exact product name and needs alternate methods to find them.

Image Search plus Text System

The image search plus text system is the first system described in the "system architecture" section. Here we try to alleviate the problem where the pure image search system can not help the user to find the desired product in the first several pages of returned results. In our evaluation, we assume that the user is presented with 50 product leaf category labels as the result of the pure image search. If the users find the correct leaf level category label for the product they want, they can just do one click on that category label to get to the right products. If the users cannot find the desired product, they need to spend one user click to navigate to the traditional text based search system and continue from there.

Our Combined Search System

In our combined search system, we use the multiple column user interface designed for the product hierarchy tree navigation. At each level, we conduct a K-Nearest-Neighbor search for all images under that product tree node to find the top 30 images that are visually similar to the query image provided by the user. The associated

System	Avg user clicks
Text Based System	5.05
Image Search Plus Text System	3.99
Our Combined Search System	1.58

Table 4.2: Average number of user clicks for different systems

product category labels are collected from these 30 product images and sorted. Thus the top candidate category label in the list receives the most votes from the product images that are visually similar to the query image. In our experiment, our system provides 10 predicted category labels at top of each column and lists the rest of the category labels after them.

Comparison of different systems

Table 4.2 shows the average number of user clicks needed to navigate to the correct leaf level product category for three different systems described above. As we can see, the traditional text based system averages the largest number of user clicks. The image search plus text system improves the amount of average user clicks by about 21%. The improvement is not significant due to the poor accuracy of image similarity search in a large-scale system. For our combined search system, we achieve the best user experience in terms of the user clicks with an additional 60% improvement over the image search plus text system.

4.6 Summary

In this chapter, we presented the design and implementation of a search system that uses content-based image search to assist user navigation of the product category tree. The system leverages the existing product hierarchical structure to compensate the weakness of state of the art computer vision technology on large datasets. Our evaluation with a real e-commerce dataset shows a 60% improvement in the amount of

user clicks needed for the user to navigate to the correct product category compared with traditional content-based image search approach.

We believe our system design can also benefit other systems beyond the field of e-commerce. The idea of using content-based search to assist navigation of a hierarchical dataset also has applications in general object recognition and classification.

Chapter 5

Rank-Based Model for Sketch

5.1 Introduction

Content-based similarity search for massive amounts of feature-rich (non-text) data has been a challenging problem because feature-rich data objects such as images, audio, and other sensor data is typically represented by many feature vectors with tens to hundreds of dimensions each. As a result, the key challenge in designing a content-based similarity search engine is solving the general high-dimensional search problem for very large datasets. In other words, we must understand how to find data objects similar to a query data object quickly with small data structures.

Although past research has made significant progress on the high-dimensional search problem, there is still no satisfactory general solution. Tree data structures such as R-tree [55], K-D tree [12], SR-tree [65], and more recently
navigating-nets [70] and cover-tree [13], have been proposed to solve the K-Nearest-Neighbor (KNN) search problem in high-dimensional spaces. But they work well only when the (intrinsic) dimensionality is relatively low. When the dimensionality is beyond 15 or so, such approaches are typically slower than the brute-force approach which scans through all data objects in the dataset. Recently, several indexing meth-

ods based on locality sensitive hashing (LSH) [60, 50, 30, 87] have been proposed for approximate KNN search, but they are also limited to relatively low intrinsic dimensionality. When the intrinsic dimensionality is high, the LSH approach typically requires hundreds of hash tables to achieve reasonably good search quality.

A promising approach is to use *sketches* as compact metadata for a similarity search engine. Sketches, which are constructed from domain-specific feature vectors, have two salient properties: their small size and the ability to estimate the distance between two feature vectors from their sketches alone. At search time, sketches are used to filter out unlikely answers, resulting in a much smaller candidate set which is then ranked with a sophisticated distance function on the original feature vectors. This approach is practical for two reasons: the first is that for feature-rich data with high intrinsic dimensionality, filtering metadata is more efficient in both space and time than other known approaches. The second is that content-based similarity search capability is often integrated with traditional search tools based on attributes such as time, location, and other annotations. A typical integration method is to perform an attribute-based search to produce an intermediate dataset which can be filtered on-the-fly efficiently into a small candidate set for final ranking.

Recent theoretical and experimental studies have shown that sketches constructed based on random projections can be used to approximate ℓ_1 distance and that such sketches can achieve good filtering accuracy while reducing the metadata space requirement and speed up similarity searches by an order of magnitude[23, 79]. An important advantage of this approach over other dimension reduction techniques is that sketch construction based on random projections requires no prior knowledge about the content of the datasets. The challenge of designing a real system using this approach is to choose the sketch size wisely.

Properly sized sketches can greatly reduce the storage requirement for metadata and speed up similarity search while maintaining good search quality. An important

design decision is sketch size in bits, given the desired filtering quality and the dataset size of a specific data type. Choose too few bits, and the distance estimates computed from the sketches will be inaccurate. Choose too many bits, and the sketches will needlessly waste storage space and CPU time. Ideally, a system designer can determine the sketch size and other parameters of the algorithm at system initialization time when he knows only the targeted data type, dataset size, and perhaps a small sample dataset. In order to achieve this goal, we need to model the relationship between the sketch size and other information and understand how to use the model in real systems designs.

This chapter presents two analytical and experimental results to help systems designers achieve the goal above. The first is a rank-based filtering model for the random projection based sketching technique that uses Hamming distance to approximate ℓ_1 distance. We have validated the model with image, audio, and 3D shape datasets and shown that the model can conservatively predict the required sketch size, given desired filtering quality, target dataset size, and filtering result size.

The second is the result of investigating how to use the rank-based filtering model to help systems designers make design decisions without the whole dataset. Experimental results on three real datasets show that the rank-based filtering model performs well, yielding useful, conservative predictions for larger datasets even though the parameters of the model are set with a small sample dataset. This result allows systems designers to build the model into a software tool.

We then show how to apply the analytical results to size sketches in configuring a content-based similarity search tool for a 3D-shape dataset. The case study shows that the analytical model is convenient to use.

5.2 Previous Work

Similarity search is typically formulated as a k nearest neighbor (KNN) search problem. The exact form of this problem suffers from the "curse of dimensionality" – either the search time or the search space is exponential in dimension d [35, 84]. Previous study [128] has shown that when the dimensionality exceeds around 10, space partition based KNN search methods (e.g. R-tree [55], K-D tree [12], SR-tree [65]) perform worse than simple linear scan. As a result, researchers have instead focused on finding approximate nearest neighbors whose distances from the query point are at most $1 + \epsilon$ times the exact nearest neighbor's distance.

Recent theoretical and experimental studies have shown that sketches constructed based on random projections can effectively use Hamming distance to approximate ℓ_1 distance for several feature-rich datasets [23, 80, 79]. A recent work shows that such sketches can be used as an efficient filter to create candidate sets for content-based similarity search [78], which focused on efficient filtering methods of data objects each represented by one or multiple feature vectors, and not on the rank-based analytical model and experimental results.

Although recent theoretical and experimental research has made substantial progress on sketch constructions for building large-scale search engines and data analysis tools [43], not much work has been done on modeling sketches. Broder did an excellent analytical analysis for sketches constructed based on min-wise permutations for near-duplicate detection [17]. Since the application is for near-duplicate detection, his method is based on probabilistic analysis for random distribution of data.

Several approximation-based filtering techniques for KNN search have been proposed in the literature. For example, the Vector Approximation file (VA-file) [128] method represents each vector by a compact, geometric approximation where each dimension is represented by l bits. Other approximation techniques such as A-tree method [96] and Active Vertice tree (AV-tree) method [9] were also proposed. Al-

though experimental results using different approximation sizes were reported for these methods, no formal analysis on how to choose the approximation parameters were given.

Most previous work on content-based similarity search of feature-rich data has focused on segmentation and feature extraction methods. We are not aware of prior work on modeling the Lp distance distributions of feature-rich data such as image, audio and 3D-shape datasets. Previous work either assume uniform distribution of feature vectors in high dimensional spaces, or present end results using the whole dataset. The notions of doubling dimension and intrinsic dimensionality (see [70, 13]) have been used previously to capture the inherent complexity of data sets from the point of view of several algorithmic problems including nearest neighbor search. However these notions do not provide a fine-grained model for distance distributions and do not have enough information to accurately estimate the performance of filtering algorithms for nearest neighbor search. By modeling distance distributions of a dataset, our analytical model can be adapted to different data types, and only a small sample dataset is needed for the analytical model to give good predictions for larger datasets.

5.3 Filtering for Similarity Search

This section describes the similarity search problem, sketching algorithm, and filtering method using sketches that are considered in our analytical and experimental study.

5.3.1 Similarity Search

Informally, similarity search refers to searching a collection of objects to find objects similar to a given query object. The objects we will be interested in are noisy, high-dimensional objects such as images and audio recordings. Here, similarity between objects refers to a human-perceived notion of similarity. This informal notion of simi-

larity search is made concrete as follows: objects are represented by high-dimensional feature vectors and similarity is defined in terms of a distance metric on the underlying space of features. Given a query object, q, in this setting, the goal is to find nearby objects, r, such that the distance $d(q, r)$ is small. In particular, we may ask for all objects r within some chosen distance of the query point, or more often, we may ask for the k nearest neighbors of the query point. This latter formulation of the search problem is commonly referred to as the k nearest neighbor (KNN) problem.

Although the choice of how to extract features and which distance function to use are domain specific, in practice, it is frequently the case that objects are represented by D-dimensional real-valued vectors and the perceptual distance between objects is modeled by one of the ℓ_p norms. For a pair of points $X = (X_1, \ldots, X_D)$ and $Y = (Y_1, \ldots, Y_D)$, these distance functions have the form:

$$d(X, Y) = \Big(\sum_{i=1}^{D} |X_i - Y_i|^p \Big)^{1/p}$$

5.3.2 L1 Sketch Construction

In this chapter, we focus on a recently proposed sketching technique [80]. The sketches constructed using this technique are bit vectors and the Hamming distance between two bit vectors approximates the weighted ℓ_1 distance between the original feature vectors. This sketching technique has proved useful in several application domains[79].

Briefly, the sketch construction algorithm works by randomly picking a threshold in a particular dimension and checking if the vector's value in that dimension is larger (bit 1) or smaller (bit 0) than the threshold. Let B be the sketch size in bits, and H be the XOR block size in bits. Each sketch is constructed by first generating $B \times H$ bits and then XORing every H consecutive bits, resulting in the final B-bit sketch. By XORing H bits into 1 bit, this algorithm produces a dampening (or thresholding) effect such that smaller distances are approximated with higher resolution, making it

suitable for nearest neighbor search[1].

Let w_i be the "weight" (or importance) assigned by the domain-specific feature extraction algorithm to dimension i and let l_i and u_i, respectively, be the minimum and maximum values for the i-th coordinate over all observed feature vectors. Let D be the dimension of the feature vector. At system startup time, $B \times H$ random (i, t_i) pairs are generated using Algorithm 1. At run time, the D-dimensional feature vector x is converted into a B-bit bit vector using Algorithm 2. For further details and a proof of correctness, we refer the reader to [80].

Algorithm 1 Generate $B \times H$ Random (i, t_i) Pairs

input: $B, H, D, l[D], u[D], w[D]$
output: $p[D], rnd\_i[B][H], rnd\_t[B][H]$

$p_i = w_i \times (u_i - l_i); \quad for\ i = 0, \ldots, D-1$
normalize $p_i \quad s.t.\ \Sigma_{i=0}^{d-1}\ p_i = 1.0$

for $(b = 0; b < B; b++)$ **do**
  for $(h = 0; h < H; h++)$ **do**
    pick random number $r \in [0, 1)$
    find $i \quad s.t.\ \Sigma_{j=0}^{i-1}\ p_i <= r < \Sigma_{j=0}^{i}\ p_i$
    $rnd\_i[b][h] = i$
    pick random number $t_i \in [l_i, u_i]$
    $rnd\_t[b][h] = t_i$
  end for
end for

5.3.3 Filtering using Sketches

Since sketches require little storage space and since the distance between query objects can be estimated from sketches efficiently, sketches can be used to implement a filtering query processor for similarity search. A filtering query processor first constructs a candidate set of result objects for a given query object on the basis of sketch distance. The candidate set size is chosen to be large enough such that it is likely

[1] See Formula 5.1 in Section 5.4 for details.

Algorithm 2 Convert Feature Vector to B-Bit Vector

input: $v[D], B, H, rnd\_i[B][H], rnd\_t[B][H]$
output: $bits[B]$

for $(b = 0; b < B; b++)$ **do**
  $x = 0$
  for $(h = 0; h < H; h++)$ **do**
    $i = rnd\_i[b][h]; \quad t_i = rnd\_t[b][h]$
    $y = (v_i < t_i \ ? \ 0 : 1)$
    $x = x \bigoplus y$
  end for
  $bits[b] = x$
end for

to contain the k-nearest neighbors under the original distance on feature vectors. In effect, the construction of the candidate set "filters out" the vast majority of objects in the system that are far from the query object while still capturing the objects close to the query. Since sketches are small and distance estimation on sketches are very efficient, a simple, yet practical approach for generating this candidate set is a linear scan through the set of all sketches.

The second step in a filtering query processor is the ranking of the candidate set by the original distance metric on the original feature vector. This exact computation need only be carried out once for each point in the candidate set. The k-nearest neighbors in the candidate set under the original distance metric is then taken as the query result set. The underlying assumption in a filtering query processor is that the k-nearest neighbors in the candidate set is an accurate estimate of the k-nearest neighbors in the full data set. In practice, one must choose the candidate set to be large enough that it captures a sufficiently large fraction of the k-nearest neighbors under the original distance, but not so large that it adversely affects search engine performance. If the candidate set is too small, the query processor will be fast, but the search quality may be poor. On the other hand, if the candidate set is too big, the processor will waste time and resources on unlikely candidates. We can capture

this inherent trade off between search quality and filter set size by asking what filter ratio is necessary to achieve a particular quality goal. If k is the number of results to return, a filter with filter ratio t will return a candidate set of size $t \times k$. A filtering query processor seeks to optimize t for a given fraction of the k-nearest neighbors in the final result set.

A system designer who adopts the filtering approach to similarity search must choose not only a particular domain-specific feature representation and distance function, but also an appropriate sketching algorithm and a set of parameters for sketching and filtering. More specifically, we are mainly interested in answering the following questions:

- What is an appropriate choice for the sketch size, B?
- How to size the sketch if the input data set grows over time?

We are also interested in other parameters involved for sketching such as the best H value for XORing and best filter ratio t when constructing sketches as they are part of the sketching parameters system designer need to decide at design phase. The rest of the chapter presents our analytical and experiment results to answer these questions.

5.4 Analytical Model

We use the following notation:

- B: sketch size in bits
- k: number of similar objects to return
- t: filter ratio – *i.e.* filtered set size is $k \times t$
- H: XOR block size in bits for sketching
- S: the set of domain-specific feature vectors

- D: the dimensionality of vectors in S
- $d(x, y)$: the domain-specific distance on $x, y \in S$
- $s^0(x)$: the $H \times B$-bit sketch of $x \in S$ before XORing
- $s(x)$: the B-bit sketch of the feature vector $x \in S$
- $d_s(x, y)$: the sketch distance between $x, y \in S$

We now describe a simple analytical model for filtering using the ℓ_1 sketch of Section 5.3.2. This model provides a basis for system designers to choose appropriate parameter values for a sketch-based filtering similarity search query processor. In particular, for a given data set size, N, and result set size, k, the model predicts the relationship between recall, filter ratio (t), sketch size (B), and XOR block size (H). Thus, the model allows a system designer to choose the system parameters in anticipation of future growth.

In the following description let S be a set of N objects, each represented by a D-dimension feature vector. Given objects q and r, let $d(q, r)$ be the *feature distance* between q and r, $s(q)$ and $s(r)$ be the sketches of q and r, respectively, and $d_s(q, r)$ the *sketch distance* between q and r. We define the *rank* of r given q to be the number of points in S that precede r when objects are ordered in increasing order of *feature* distance from q. For a fixed query q, let r_i denote the ith object in S in this ordering. Similarly, we define the *sketch rank* of r to be the number of points in S that precede r when objects are ordered in increasing order of *sketch* distance to q.

The goal is for the analytical model to answer the following question: Given N, k fixed, as a function of t, B, and H, what fraction of the points $p \in S$ with rank at most k have sketch rank at most $k \times t$? We develop the model in a series of steps. First, we describe how we model the distribution of feature distances in the data set. Second, we obtain an expression for the distribution of the sketch distance as a

function of the feature distance. Next, we model the distribution of the sketch rank of an object $r \in S$ for a query q as a function of its feature distance from q. This uses the distribution of feature distances in the data set and distribution of sketch distances. Finally, we use this model for the sketch rank to estimate the recall for a given filter ratio value. Each of these steps is described in the subsections that follow.

5.4.1 Distance Distribution

Since the sketch distance between two objects is related to the original feature vector distance, we first study the distribution of feature vector distances. For one particular query object, we calculate the feature vector distances of all the other objects in the dataset to this query object. The histogram of all the object feature distances forms the feature vector distance distribution for that particular query object. With the feature distance distribution known, we will be able to predict the sketch distance between the query object and rest of the objects using the analytic model described in the next section. Note that in k-nearest neighbor search, objects that are nearby have much more impact on the overall search quality than the ones that are further away. When we model the distance distribution, we are mostly interested in the distance distribution close to the k nearest neighbors.

One of the goals for our approach is to predict the sketch performance when the dataset size changes. In order to do this, we predict the distribution of object distances in a data set using the distribution of distances in a smaller sample. The basis for this is the hypothesis that every data type is associated with an underlying object distance distribution. The particular distances observed in a specific data set can be viewed as a sample of the distribution associated with the data type. For example, in Figure 5.1, we compare the average distance distribution of 100 query points with the full dataset with that of a uniformly sampled dataset with only one-tenth of the data points.

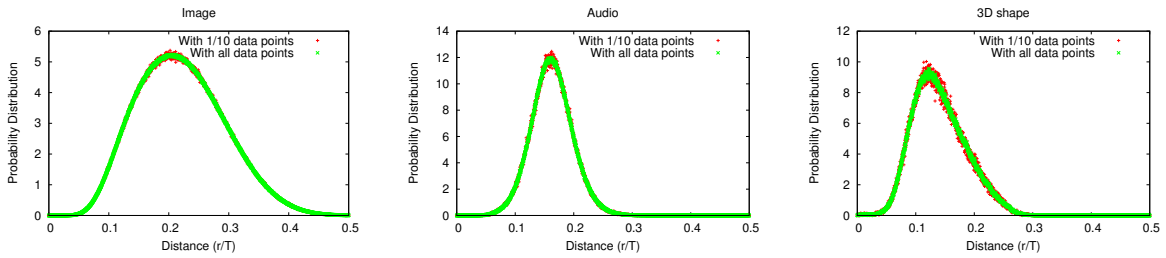


Figure 5.1: Compare Distance Distribution of Full Dataset and 1/10 of Dataset.

One subtlety in modeling distances is that the distribution of distances from different query objects can be different and using a single distribution for them can lead to errors. The distance distributions for different query objects have similar shapes but are peaked at different points. Since our data objects have a natural bound on each dimension, the objects are contained in a high dimensional rectangle. The location of the query object in this high dimensional rectangle will affect the peak of the feature distance distribution. In order to model this variation, we pick a random sample of 100 query objects and use their distance distributions to approximate the overall distance distributions. We compared this approach with using a single average distance distribution. The latter did not perform as well as the approach that explicitly models the variation in object distance distributions.

Further, we approximate the empirical individual query object distance distributions by a distribution with a closed form expression. Due to the nature of k-nearest neighbor search, we are not trying to approximate the full distance distribution. Instead, only the distance distribution close to k-nearest neighbors are considered during fitting. The details of this appear in Section 5.6.1.

5.4.2 Sketch Distance Distribution

Given a dataset S, let w_i, u_i, l_i be the weight, upper bound and lower bound of the i-th dimension, respectively. Let $T = \sum_i w_i \times (u_i - l_i)$. Using the sketch algorithm of Section 5.3.2, for every object $r \in S$, we construct the initial bit vector $s^0(r)$ of

length $B \times H$. For a fixed query point q, consider object $r \in S$ and let $x = d(q,r)/T$. The probability that $s^0(q)$ disagrees with $s^0(r)$ in the j-th bit is:

$$\Pr[s_j^0(q) \neq s_j^0(r)] = d(q,r)/T = x$$

After XORing contiguous H-bit blocks of s^0 to produce the final B-bit sketch, the probability that the two sketches differ in bit j is:

$$\Pr[s_j(q) \neq s_j(r)] = p(x) = \frac{1}{2}\left(1 - (1-2x)^H\right) \tag{5.1}$$

Thus, the probability that the two B-bit sketches $s(q)$ and $s(r)$ differ in exactly b bits is given by the binomial distribution:

$$\Pr[d_s(q,r) = b] = p(x,b) = \binom{B}{b} p(x)^b \left(1 - p(x)\right)^{B-b} \tag{5.2}$$

where $p(x)$ is given by equation (5.1). This formula gives the probability distribution of the sketch distance as a function of the feature distance. The proof in detail can be found [80].

5.4.3 Rank Distribution

Consider an object $r \in S$. We would like to estimate the sketch rank of r, *i.e.* the number of objects that precede r when we order all objects in S in increasing order of sketch distance to query object q. A key assumption in this calculation is that the sketch distances are independent of each other. While this assumption is not completely accurate, it is a reasonable approximation. As we discuss later, this leads to a conservative estimate on the quality of the filtering results. We also assume that in the ordering by sketch distances, objects with the same sketch distance are ordered randomly, that is, for two objects with the same sketch distance, the probability that

one precedes the other is exactly 1/2.

The sketch rank of r is dependent on the sketch distance $d_s(q,r)$. Consider the event $d_s(q,r) = b$. Note that the probability of this event is a function of the feature distance $d(q,r)$ and is calculated in (5.2). Consider an object $r' \in S$ such that $d(q,r')/T = x$ and let $s = d_s(q,r')$ be the sketch distance of r'. Let $P(x,b)$ be the probability that r' is ranked lower (*i.e.* closer to q) than r when $d_s(q,r) = b$. Note that this is a function of $x = d(q,r')/T$ and the value b of $d_s(q,r)$.

$$\begin{aligned} P(x,b) &= \Pr[s < b] + \frac{1}{2}\Pr[s = b] \\ &= \sum_{i=0}^{b-1} \Pr[s = i] + \frac{1}{2}\Pr[s = b] \qquad (5.3) \\ &= \sum_{i=0}^{b-1} p(x,i) + \frac{1}{2}p(x,b) \qquad (5.4) \end{aligned}$$

Let $\text{rank}\,(r)$ denote the sketch rank of r. $\text{rank}\,(r)$ is the sum of indicator random variables $Y(r_i, r)$, one for every object $r_i \in S$. The indicator variable $Y(r_i, r)$ for $r_i \in S$ corresponds to the event that r_i precedes r in the ordering by sketch distance. Our independence assumption implies that given a value for $d_s(q,r)$, all these variables are independent. Let $x_i = d(q,r_i)/T$. Note that $\Pr[Y(r_i,r) = 1 | d_s(q,r) = b] = P(x_i, b)$ computed in (5.4). The expected value and variance of $\text{rank}\,(r)$ are given by

$$\begin{aligned} \mathrm{E}[\text{rank}\,(r)\,|d_s(q,r) = b] &= \sum_{i=1}^{N} P(x_i, b) \\ \mathrm{Var}[\text{rank}\,(r)\,|d_s(q,r) = b] &= \sum_{i=1}^{N} P(x_i, b) - [P(x_i, b)]^2 \end{aligned}$$

When we use the feature distance distribution model, let $f(x)$ to be the probability density function for the distances, *i.e.* $\int_{x_1}^{x_2} f(x)dx$ is the fraction of points $r' \in S$ such that $d(q,r')/T \in [x_1, x_2]$. We can replace the summation over all data points with an

integration over the distance distribution:

$$\mathrm{E}[\mathrm{rank}\,(r)\,|d_s(q,r)=b] = N\int_0^1 P(x,b)f(x)dx$$
$$\mathrm{Var}[\mathrm{rank}\,(r)\,|d_s(q,r)=b] = N\int_0^1 (P(x,b)-P(x,b)^2)f(x)dx$$

Given the fact that N is usually on the order of hundreds of thousands, the distribution of $\mathrm{rank}\,(r)$ (for a specific value of $d_s(q,r)$) is approximately normal by the Central Limit Theorem. The normal distribution parameters can be determined by $\mathrm{E}[\mathrm{rank}\,(r)\,|d_s(q,r)=b]$ and $\mathrm{Var}[\mathrm{rank}\,(r)\,|d_s(q,r)=b]$. Thus the probability that $\mathrm{rank}\,(r)$ is at most M can be expressed as:

$$\Pr[\mathrm{rank}\,(r_k)\le M|d_s(q,r)=b] = \int_0^M f(y;\mu_b,\sigma_b)dy$$

where

$$\mu_b = \mathrm{E}[\mathrm{rank}\,(r)\,|d_s(q,r)=b]$$
$$\sigma_b = \sqrt{\mathrm{Var}[\mathrm{rank}\,(r)\,|d_s(q,r)=b]}$$
$$f(y;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-(y-\mu)^2/2\sigma^2}$$

Now, we can write the distribution of $\mathrm{rank}\,(r)$ as a mixture of normal distributions, one for each value of $d_s(q,r)$. The distribution for b is weighted by $\Pr[d_s(q,r)=b]$. This gives us the distribution of $\mathrm{rank}\,(r)$ and allows us to calculate the probability that $\mathrm{rank}\,(r)$ is at most M as follows:

$$\Pr[\mathrm{rank}\,(r)\le M] = \sum_{b=0}^{B}\Pr[d_s(q,r)=b]\int_0^M f(y;\mu_b,\sigma_b)dy$$

We overload the notation somewhat and use $\mathrm{rank}\,(x)$ for $x\in[0,1]$ to denote the sketch rank of an object $r\in S$ such that $d_s(q,r)/T = x$. Note that $\Pr[d_s(q,r)=b] =$

$p(x, b)$. Using the previous expression for $\Pr[\text{rank}\,(r) \leq M]$, we get

$$\Pr[\text{rank}\,(x) \leq M] \;=\; \sum_{b=0}^{B} p(x,b) \int_0^M f(y; \mu_b, \sigma_b) dy$$

Given this expression for the rank distribution, we can now estimate search quality for a given filter set size, M.

5.4.4 Search Quality Estimation

Once we have an expression for the rank distribution for objects $r \in S$, for a given filter set size M, the expected fraction of the k nearest neighbors being included in the filtered set (*i.e.* the recall) can be computed as:

$$\text{Recall} = \frac{1}{k} \sum_{j=1}^{k} \Pr[\text{rank}\,(r_j) \leq M]$$

When the feature distance distribution is used, the recall can be calculated as:

$$\text{Recall} = \frac{N}{k} \int_0^{x_0} \Pr[\text{rank}\,(x) \leq M] f(x) dx$$

where x_0 can be derived from:

$$k = N \int_0^{x_0} f(x) dx$$

Note that the sketch distributions and rank distributions are computed based on a single query object q. As a result, the search quality estimate (recall) may vary for different query points. To ensure that the results are representative of the entire data set, we use multiple representative query objects to model the distance distribution. To estimate overall search quality, we average the recall value computed using the distance distributions for each of these query objects.

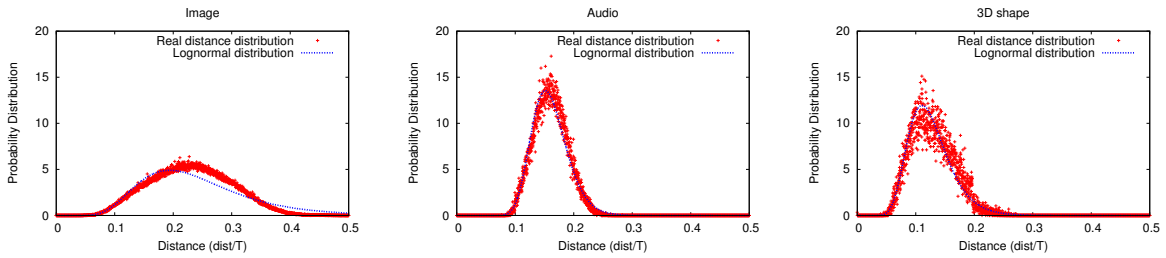


Figure 5.2: Compare the Real Distance Distribution with Lognormal Distribution: We only fit the initial part of real distribution for k-nearest neighbor search.

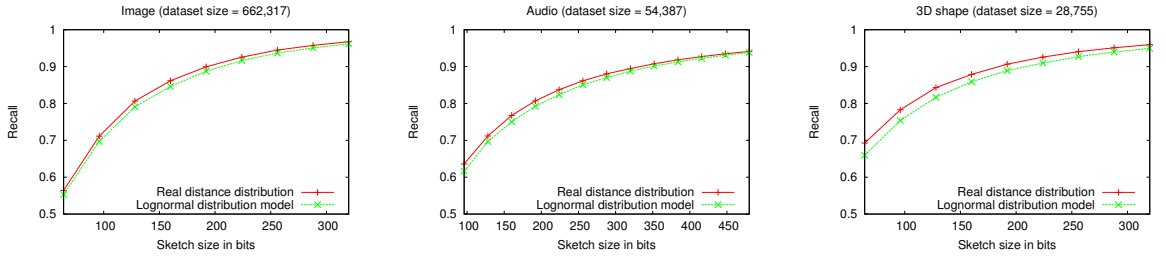


Figure 5.3: Filter Quality with Different Distribution Models (H = 3): Lognormal distribution gives conservative filtering quality prediction.

5.5 Evaluation

We have employed three kinds of feature-rich datasets to validate our models. To evaluate the filtering quality, we have used average recall in which the "gold standard" for comparison is the results computed with the original distance function.

5.5.1 Datasets

We have studied three kinds of data: images, audio, and 3D shapes. Table 5.1 provides a summary of the dataset sizes and the number of dimensions in the domain-specific feature vector representations.

Dataset	Number of Feature Vectors	Dimension
image	662,317	14
audio	54,387	192
3D shape	28,775	544

Table 5.1: Dataset Sizes and Dimensions.

Image Data

The image dataset used in our study is drawn from the Corel Stock Photo Library, which contains about 60,000 images. The main reason to choose this dataset is that it has become a standard dataset for evaluating content-based image retrieval algorithms.

We used the JSEG [34] image segmentation tool to segment each image into multiple homogeneous regions based on color and texture. On average, each image is segmented into 10 regions, resulting in about 660,000 regions in total. The image region representation we use is similar to the one proposed in [80], where each region is represented by a 14-dimensional feature vector: nine dimensions for color moments and five dimensions for bounding box information. The bounding box is the minimum rectangle covering a segment and is characterized by five features: aspect ratio (width/height), bounding box size, area ratio (segment size/bounding box size), and region centroids. The similarity between two regions is determined by weighted ℓ_1 distance on their 14-dimensional feature vectors.

Audio Data

Our audio dataset is drawn from the DARPA TIMIT collection [46]. The TIMIT collection is an audio speech database that contains 6,300 English sentences spoken by 630 different speakers with a variety of regional accents. We chose this dataset also because it is available to the research community.

We break each sentence into smaller segments and extract features from each segment. For each audio segment, we use the Marsyas library [119] to extract feature vectors. We begin by using a 512-sample sliding window with variable stride to obtain 32 windows for each segment and then extract the first six MFCC (Mel Frequency Cepstral Coefficients) parameters from each window to obtain a 192 dimensional feature vector for each segment. We use weighted ℓ_1 distance on the 192-dimensional

feature vectors to determine similarity. As a result, the sentences of the dataset are partitioned into about 54,000 word segments, and we extract one feature vector per word segment.

3D Shape Models

The third dataset we use in our study contains about 29,000 3D shape models, which is a mixture of 3D polygonal models gathered from commercial viewpoint models, De Espona Models, Cacheforce models and from the Web. Each model is represented by a single feature vector, yielding about 29,000 feature vectors in total.

To represent the 3D shape models, we use the Spherical Harmonic Descriptor (SHD) [66], which has been shown to provide good search quality for similarity search of 3D shape models. The models are first normalized, then placed on a $64 \times 64 \times 64$ axial grid. Thirty-two spheres of different diameters are used to decompose each model. Up to order 16 spherical harmonic coefficients are derived from the intersection of model with each of the 32 spherical shells. By concatenating all the spherical descriptors of a 3D model in a predefined order, we get a $32 \times 17 = 544$-dimensional shape descriptor for each 3D model. Although the original SHD algorithms used ℓ_2 distance as the similarity metric, we have found that ℓ_1 distance delivers similar search quality. As a result, in this study we use ℓ_1 distance on the 544-dimensional feature vector.

5.5.2 Evaluation Metrics and Method

We have conducted two types of experimental studies in this chapter: distribution model fitting and filtering model validation.

For distribution model fitting, we use some common distribution functions to fit the distance distribution and compare the fitted distance function with the real distribution. In order to validate the fit, we compare the residuals after the least

squared fitting and also plot the result for visual inspection. Moreover, we put each fitted distribution function into our model and compare their results with the result using real distance distribution to determine the best distribution function.

For filtering model validation, we compare the filtering results predicted by the model with those by an implementation of the sketch-based filtering algorithm. The filtering qualities are measured against the *gold standard* of each dataset, which are computed by a brute-force approach over the entire dataset using the original distance function. Specifically, we compare the recall value at filter ratio t predicted by our model with that computed experimentally. The recall at filter ratio t is the fraction of the k nearest neighbors to the query point that appear in the first $t \times k$ objects found by the filtering search algorithm. An ideal filtering algorithm will have a recall value of 1.0 at a filter ratio of 1. In practice, a good filter is one that achieves a satisfactory recall with a small filter ratio. Since re-ranking of the candidate objects filter using the original feature vectors is done after filtering, we do not need to report the precision of our filtering method here.

The method used in our experimental evaluation is to pick one hundred objects uniformly at random from each dataset as queries. For each query object, we use the domain-specific feature vector distance to compute the k nearest neighbors. We then fix the size of the sketch produced by the sketching algorithm and for each object in the dataset generate a sketch of that size, and use the sketches to compute the filtered candidate set of $t \times k$ objects and calculate the fraction of the k nearest neighbors appearing in this set. Since the sketching algorithm is itself randomized, we take the average recall over ten instances of the sketch algorithm. Finally, for each sketch size, we report the average recall value at a fixed filter ratio t over the one hundred randomly chosen query objects and compare that recall to the recall predicted by our model.

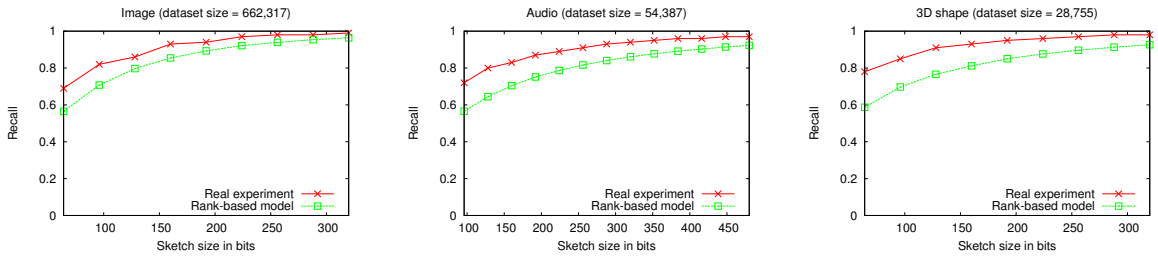


Figure 5.4: Filter Quality vs Sketch Size (H = 3): Model gives conservative estimate, and the estimate is closer when recall value is high

5.6 Experimental Results

We are interested in answering the following three questions:

- How shall we model the distance distribution of real datasets to be used in the analytical model?
- How well can the analytical model help system designers choose design parameters such as sketch size?
- How well can the analytical model predict for large dataset if its parameters are set with a small sample dataset?

This section reports the experimental answers to these questions.

5.6.1 Distance Distribution Model

In this section, we explore the possibility of using a simple distribution – with closed form expression – to model the real data distance distribution. This allows us to model the system with fewer parameters (typically two) rather than the full distance distribution. Furthermore, we can reuse the distribution model when the dataset size grows.

We have investigated several common distance distributions which may be used to model the observed real distance distribution from the dataset and decided to choose

lognormal distribution in our experiments. Note that in k-nearest neighbor search, objects that are much further away than the k-th nearest neighbor have much less impact on the overall search quality than the ones that are closer. As a result, the distance distribution close to the k nearest neighbors are the most important in the overall result.

Based on this observation, we use the distance values of the $2 \times k \times t$ closest data points to fit the statistical models, and then use the models to extrapolate to the full set of distances. We use GNU Scientific Library [39]'s nonlinear least-square fitting library to find fitting parameters. We only use initial part of the distance distribution corresponding to the $2 \times k \times t$ closest data to do the fitting. It tries to minimize the error between the real distance distribution and the corresponding portion of lognormal distance distribution. As shown in Figure 5.2, the lognormal distribution fits the data, even when extrapolated to the full dataset. Since this shows the distance distribution of just a single query point, we can see the distance distribution is not as smooth as in Figure 5.1 where the average distance distribution of 100 query points is used.

To model the real dataset, we use 100 randomly chosen query points to generate 100 distance distributions. After that, each distance distribution is fitted with the lognormal distribution. We then use these 100 sets of lognormal distribution parameters to model the distance distribution of the whole dataset. This helps us to model the variation of distance distributions as seen from different data points in the real dataset.

We have also validated the choice of the distribution model by comparing the filtering result generated from the real distance distribution with that generated by the model distribution. Figure 5.3 shows the filtering results using lognormal distribution model, together with the filtering results when the real distance distribution is used[2].

[2]The real distance distribution is directly computed from the dataset and no parameter fitting is performed.

We can see that the lognormal distribution generates close trend to the real distance distribution. It is important to note that using lognormal distribution gives a conservative estimate of the recall compared to the real distribution. From the system designer's perspective, this is desirable behavior since the recall is bounded by the model at a small cost in sketch size.

5.6.2 Sizing Sketches

The goal of the analytical model is to help systems designers choose design parameters properly. The following reports our experimental results to see how well the model can help systems designers choose sketch size B and the related parameter XOR Bits H properly. In our experiments, we set the result set size k to be 100 and filter ratio t to be 10.

Choosing sketch size B Sketch size in bits B is perhaps the most crucial parameter in a sketch-based filtering mechanism for similarity search. Finding a good sketch size for a real system with a given expected dataset size will deliver high filter quality with minimal storage requirement.

Figure 5.4 shows the trend of filtering quality for different sketch sizes. The recall at a filter ratio of 10 is used to compare the experimental result with our analytical model. As expected, using more bits in a sketch leads to higher filtering quality.

Suppose we wish to build a system that achieves a recall of 0.9. Our results show that the sketch sizes must be about 160 bits, 256 bits and 128 bits for the image, audio, and 3D shape datasets, respectively. The amount of storage required for sketch storage are substantially smaller than the feature vector storage requirement. We use these sketch sizes in the following experiments.

Notice that our analytical model conservatively predicts the average recall in all cases and the predicted trend is consistent with the experimental results. This implies

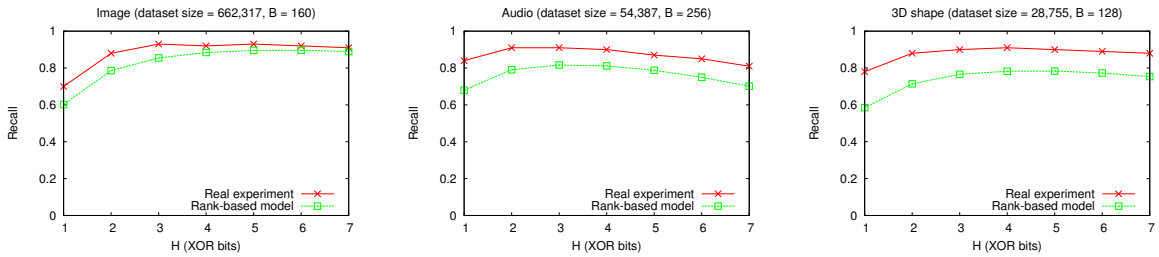


Figure 5.5: Filter Quality vs XOR Block Size H: Model shows same trend as real experiments, quality is good around H=3 and is not sensitive beyond that

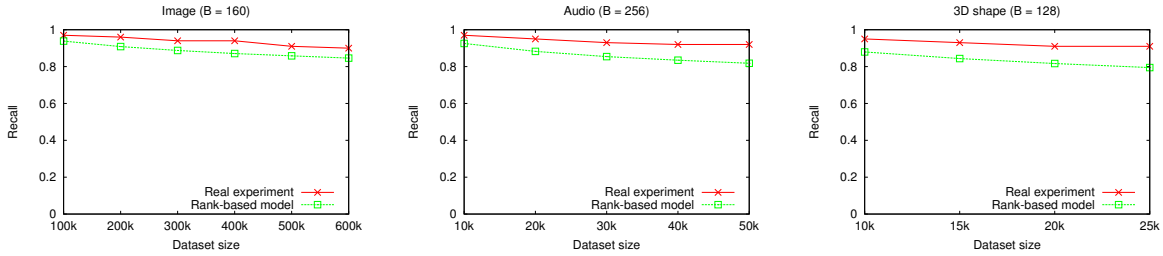


Figure 5.6: Filter Quality vs Dataset Size (H = 3): Model gives consistent and conservative prediction of quality as dataset grows

that the model is appropriate for conservative estimates for real systems designs. We will discuss the reasons for the consistent underestimation in Section 5.6.4.

Choosing XOR Bits H Choosing the H judiciously is important because a good choice of H may give better filtering quality without increasing the sketch size and thus the storage requirement. Although the best H value is data type dependent, our analytical model can help choose the value without experimenting with full original dataset. We found that the best H value is relatively stable when the sketch size B changes, so in practice we will choose the best H value first.

Figure 5.5 shows that our analytical model predicts similar H values to the experimental results. For the image dataset, both predicted and experimental results indicate the the best H value is 3. For audio dataset, the model predicts that the best H value is 3 and the experimental results show that the best is 2. For 3D shape data both indicate that the best H value is 4.

5.6.3 Extrapolating to Larger Dataset Size

When building a real system, it is common not to have the full dataset available at the initial deployment. It is important to be able to choose system parameters with only a small sample dataset, and have some performance and quality guarantees as the dataset size grows. Our analytical model is useful in this scenario since the system designer cannot conduct full-scale experiments to figure out the parameters to be used.

In order to validate our model's prediction, we conducted an experiment that simulates dataset growth. For each dataset, we used a small subset of the full dataset to configure our model: that is, we only use one tenth of the total data objects to model the distance distributions and then use the model parameters derived from small dataset to predict the filter quality when dataset size grows. The result is compared with the experimental results, where more and more data points in the dataset are included in each experiment to simulate the growing dataset.

Figure 5.6 shows the filtering quality with different dataset sizes. In each plot, the first data point corresponds to the small sample dataset that we use to derive our model parameters; the following data points labeled as "rank-based model" are the projected results using the model. The experimental results are also shown in the same plot.

The results show that the filtering quality degrades gradually as the dataset grows larger. The model can give a good prediction on the degree of quality degradation as the dataset size grows. The prediction works better when the sample dataset size is reasonably large as seen in the image dataset. For other datasets, the degradation prediction is more conservative, but conservative estimates are more acceptable than optimistic in real systems designs.

5.6.4 Discussion

For all the figures showing the recall value, we noticed a consistent underestimate of the model result compared with the experimental result. In fact the underestimate of the model is largely due to the simplified independence assumption of the model – *i.e.* the assumption the sketch distances of objects are independent of the k-th nearest neighbor's sketch distance.

In section 5.4.3, we assumed that the sketch distances for different objects $r \in S$ are independent This assumption simplifies the model, but in reality, there is some dependency that the model ignores. In fact, when r's sketch distance is large, the other sketch distances are also likely to be large and vice versa. In other words, there is a small positive correlation between sketch distances. In order to understand how such a correlation arises, it is instructive to consider the 1-dimensional case. Consider objects r and r' such that $d(q, r) \leq d(q, r')$. The algorithm to generate the bit vector sketch picks random thresholds for a dimension and checks if the coordinate in that dimension is above or below the threshold. The bit thus generated for q and r is different if the random threshold separates q and r. If this happens, it also likely to separate q and r'. If the sketch distance of r is large, then q and r must have been separated by several such randomly picked thresholds. But then it is likely that q and r' are also separated by these thresholds. Thus, the sketch distance of r' is likely to be large.

The positive correlation between sketch distances results in the rank of r_k being lower than that predicted by the independent model. In order to understand this, consider the extreme situation where the positive correlation is of the following form: for i randomly chosen in $[0, 100]$, each sketch distance is equal to the value at the i-th percentile of its individual distribution. In this case, objects with higher feature distance than the kth nearest neighbor r_k will never have their sketch distance lower than the sketch distance of r_k. The effect of positive correlation is similar, but less

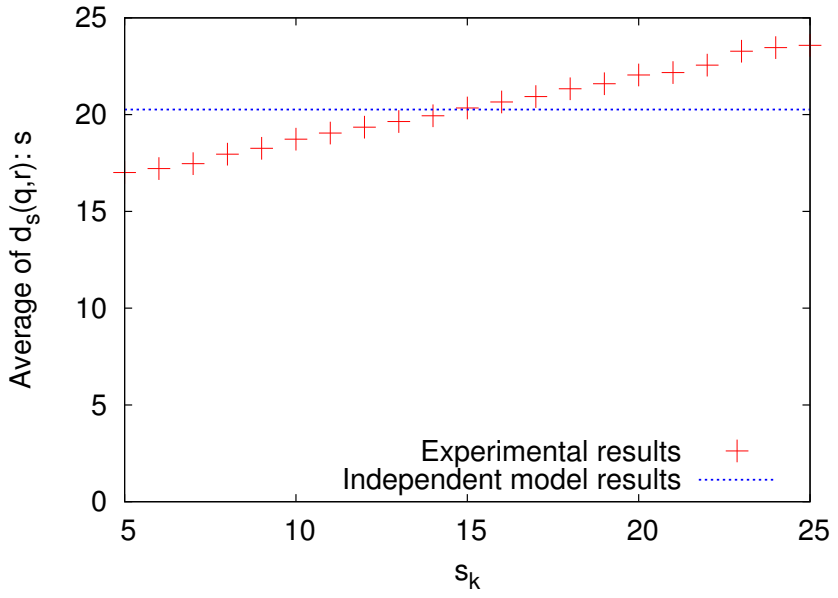


Figure 5.7: Dependence of s to s_k

extreme than the situation described above. In other words, the positive correlation lowers the probability that objects further than the kth nearest neighbor will have their sketch distance lower than the sketch distance of r_k.

We conducted an experiment with the real dataset which clearly demonstrates such a dependence. In the experiment, we repeated the sketch construction 100,000 times and observed the relationship between sketch distance s of a particular data point r and the sketch distance s_k of the kth nearest neighbor r_k. Figure 5.7 shows the result. The points show the average value of s when s_k takes different values and the dashed line shows the constant s value expected with independent model. We can see that there is a small positive correlation between r's sketch distance s and r_k's sketch distance s_k. Figure 5.8 further shows the experimental result where the (empirically observed) probability distributions of r's sketch distance is plotted for two different values of s_k.

This data dependence affects the expected probability of r overtaking r_k. The experimental result shows that the probability of that particular data point r overtaking r_k is 0.125 while our independent model's prediction is 0.167 according to

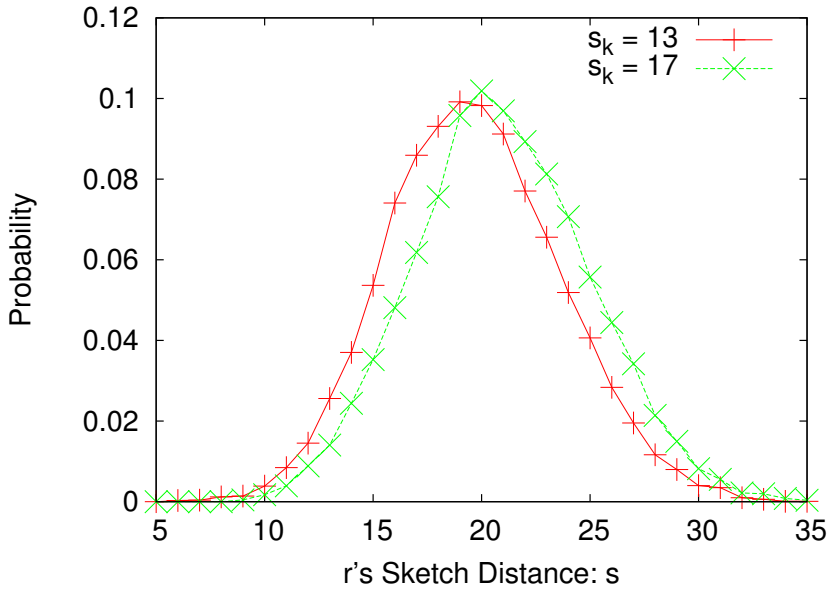


Figure 5.8: Probability Distribution of s with Different s_k

Equation 5.4. The higher rank prediction of r_k's sketch distance of our model will generate lower recall value in the quality score at filter ratio t and cause a consistent underestimate to the experimental results.

In order to accurately model the dependence of data object r's sketch distance r on r_k's sketch distance r_k, much more information about the data set is needed: value distributions on each dimension, data value dependence between different objects on each dimensions, *etc.* While this might give more accurate predictions, it is much harder to obtain reliable estimates of such fine grained information about the data set. Also it is unclear how well a model that incorporates such detailed information can be extrapolated to larger data set sizes. We have decided to adopt a simpler model that captures the essence of the experiment. Although our model gives a consistently low estimate of recall, it matches the general trend of the experimental results well.

5.7 Case Study

In this section, we use the 3D shape dataset as an example to illustrate how to use the rank-based analytical model to decide the parameters for sketching.

The sample dataset consists of 10,000 3D shape models, each represented by a 544-dimensional feature vector. The first step is to compute the feature vector distance distribution by randomly picking a number of query vectors, for instance 100 of them, and computing their ℓ_1 distances to the feature vectors in the sample dataset. As shown in Figure 5.1, the feature distance distribution of the sample dataset is similar to that of the larger dataset.

Next, we use nonlinear least-squares curve fitting to find the parameters of the lognormal distribution that best approximate the feature distance distribution we have computed. The results are shown in Figure 5.2 and Figure 5.3. This gives us a closed form distance distribution which we can then use in the rank-based filtering model.

Next, using the analytical model of Section 5.4, we can compute the estimated filtering quality for a target dataset size – in this case 28,755 objects – using different sketch sizes B (Figure 5.4) and different XOR block size H (Figure 5.5).

Based on the estimations of the analytical model, we can then decide the sketching parameters. For example, suppose we want to design a system with a target recall value of around 0.8. Then the model predicts a sketch size of 128 bits and a XOR block size of 3 bits. Given that the analytical model's predictions are conservative, we know that a similarity search system using $B = 128$ and $H = 3$ will be able to achieve recall value greater than 0.8 in practice and the sketches are $544 \times 32/128 = 136$ times smaller than the original shape descriptors[3]. If we want to design a system with recall value around 0.9, we can also consult the model to pick a sketch size of 256 bits and

[3]We assume each of the 544-dimensions of a shape descriptor is represented by a 32-bit floating point number.

XOR block size of 3 to achieve the desired quality. The sketches are $544 \times 32/256 = 68$ times smaller than the original shape descriptors.

5.8 Summary

This chapter reports the results of modeling the parameters of using sketches to filter data for similarity search. The goal of our study is to help systems designers choose key design parameters such as sketch size. We validated our model with three feature-rich datasets including images, audio recordings, and 3D shape models. Our study shows two main results for sketches that use Hamming distance to approximate ℓ_1 norm distance:

- We have proposed a rank-based filtering model for the sketch construction to use Hamming distance to approximate ℓ_1 distance. We have shown, by experimenting with image, audio, and 3D shape datasets, that this model can conservatively predict the required sketch size for required recall, given the dataset size and its filtering candidate set size.

- Using the distance distribution with its parameters derived from a small sample dataset, we show that the rank-based filtering model can be used to perform good predictions of sketch sizes for a large dataset.

Our experimental studies show that the rank-based filtering model predicts results close to experimental results. Although there are noticeable gaps between the predicted and experimental results for certain systems parameters, the predicted trends are consistent with the experimental results. Furthermore, the predictions from the model are consistently conservative in all cases.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, I studied the complex problem of how to leverage multimodal data exhibited from non-text data in order to improve similarity search system. I used three similarity search systems to study this problem. With these systems, I studied different ways of combining multiple features from multimodal data. I also introduced a rank-based model to help system builders to construct large-scale multimodal similarity search systems. Here is a list of my contributions:

- With VFerret system, I showed how to combine visual and audio features to perform effective personal video searches. VFerret system improved search accuracy from an average precision of 0.66 to 0.79.

- With Image Anti-Spam System, I explored several aggregation methods to integrate multiple image spam filters to detect image spams. The prototype system achieved a detection rate of 89% similar to traditional computer vision methods, while substantially reducing the false positive rate from around 1% to less than 0.001%. The system is also extensible, and can incorporate new image filters to counter new kinds of image spam threats.

- With Product Navigation System, I studied how to combine text search with image similarity search to help users find desired products. By allowing users to iteratively navigate down the product tree hierarchy with the aid of combined text and image similarity search, the system can reduce the number of user clicks by 60% compared to traditional methods.

- I also studied a rank-based model to help system designers construct more efficient large-scale multimodal systems. By modeling the dataset using a small subset of sample data, my algorithm can help system designers not only to determine the best parameters to create sketch, but also to predict the accuracy of similarity search results for large systems.

Although we have not found the general solution to using multimodal data in a similarity search system, this dissertation shows that it is possible to substantially improve search accuracy and efficiency by combining domain specific knowledge of multimodal data. When the similarity search system is designed carefully and case by case, it can take full advantage of multiple modalities that are intrinsic in all kinds of multimedia non-text data. My results are encouraging and point us to further research in this area.

6.2 Future work

The content-based similarity search with multimodal data is a relatively young field. Yet we have seen significant improvements in the past ten years. With new user generated digital multimedia data growing exponentially, there will be even more real world applications emerging for daily use.

The future work for large-scale similarity search systems with multimodal data are summarized in the following three directions.

- We need to improve the quality of the individual features, *i.e.*, to close the "semantic gap": Our constructed systems need to better "recognize" which contents are similar from a human's perspective. Conceptual similarity as perceived by human beings can be very different from similarity at lower data representation level (e.g. pixel level). The community needs to investigate better high quality features.

- We need to gain experience by designing more similarity search systems in order to better understand how to effectively use multiple features together: for example, emerging features such as location based features, social network based features and user generated tag features may help the overall search process generate more relevant results to satisfy user needs.

- We need to research new methods to handle large-scale multimodal data. It is a challenge to optimally combine multimodal features together using large training datasets. The current state of art training methods can only be used on small training data. As the amount of data grows exponentially, we gain access to huge amounts of training data. How to leverage such data in designing multimodal search systems is still an open question.

Content based similarity search with multimodal data is a very exciting field and I expect there will be major breakthroughs in the next ten years. Meanwhile, the real world applications for similarity search systems with multimodal data will flourish and start to impact every day life soon. These systems will help people to organize, search and benefit from the enormous amount of digital data available.

Bibliography

[1] *The DEJA VIEW CAMWEAR.* `http://www.mydejaview.com/`.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 20–29, New York, NY, USA, 1996. ACM.

[3] Khaled Alsabti. An efficient k-means clustering algorithm. In *In Proceedings of IPPS/SPDP Workshop on High Performance Data Mining*, 1998.

[4] Amazon Inc. *Amazon Mobile App.* `http://www.amazon.com/gp/anywhere/sms/bbapp`.

[5] J. andalic and, N. Campbell, S. Dasiopoulou, and Y. Kompatsiaris. An overview of multimodal video representation for semantic analysis. In *Integration of Knowledge, Semantics and Digital Media Technology, 2005. EWIMT 2005. The 2nd European Workshop on the (Ref. No. 2005/11099)*, pages 39 –45, 30 2005-dec. 1 2005.

[6] Hrishikesh B. Aradhye, Gregory K. Myers, and James A. Herson. Image analysis for efficient categorization of image-based spam e-mail. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, 2005.

[7] Stefania Ardizzoni, Ilaria Bartolini, and Marco Patella. Windsurf: Region-based image retrieval using wavelets. In *DEXA Workshop*, pages 167–173, 1999.

[8] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 6–, New York, NY, USA, 2004. ACM.

[9] Sören Balko, Ingo Schmitt, and Gunter Saake. The active vertice method: A performance filtering approach to high-dimensional indexing. *Elsevier Data and Knowledge Engineering (DKE)*, 51(3):369–397, 2004.

[10] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, RANDOM '02, pages 1–10, London, UK, 2002. Springer-Verlag.

[11] Régis Behmo, Paul Marcombes, Arnak Dalalyan, and Véronique Prinet. Towards optimal naive bayes nearest neighbor. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 171–184, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] J. L. Bentley. K-D trees for semi-dynamic point sets. In *Proc. of the 6th Annual ACM Symposium on Computational Geometry (SCG)*, pages 187–197, 1990.

[13] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Lea rning (ICML)*, pages 97–104, Pittsburgh, PA, USA, June 2006.

[14] Jinbo Bi, Tong Zhang, and Kristin P. Bennett. Column-generation boosting methods for mixture of kernels. In *Proceedings of the tenth ACM SIGKDD*

international conference on Knowledge discovery and data mining, KDD '04, pages 521–526, New York, NY, USA, 2004. ACM.

[15] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[16] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. CoPhIR: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627v2, 2009.

[17] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symp. on Combinatorial Pattern Matching*, pages 1–10. Springer-Verlag, 2000.

[18] A. J. Butte and I. S. Kohane. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Pac. Symp. Biocomput*, pages 418–29, 2000.

[19] The CAPTCHA Project, 2000. `http://captcha.net`.

[20] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proc. of the 3rd Int. Conf. on Visual Information and Information Systems*, pages 509–516, 1999.

[21] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668 –696, april 2008.

[22] Amit Chakrabarti and Graham Cormode. A near-optimal algorithm for computing the entropy of a stream. In *In ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335, 2007.

[23] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of the 34th Annual ACM Symp. on Theory of Computing*, pages 380–388, 2002.

[24] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[25] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *Proc. of the 35th Annual ACM Symp. on Theory of Computing*, pages 30–39, 2003.

[26] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the International Conference Intelligent System Molecular Biology*, 2000.

[27] Commtouch Inc. *2006 spam trends report: Year of the zombies*. `http://www.commtouch.com/documents/Commtouch_2006_Spam_Trends_Year_of_the_Zombies.pdf`.

[28] D. Sinitsyn. *Davis Handysaw software*. `http://www.davisr.com/`.

[29] Sarat C. Dass, Karthik N, and Anil K. Jain. A principled approach to score level fusion in multimodal biometric systems. pages 1049–1058, 2005.

[30] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry(SCG)*, pages 253–262, 2004.

[31] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40:5:1–5:60, May 2008.

[32] Distributed checksum clearinghouse, march 2007. `http://www.rhyolyte.com/dcc`.

[33] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 71–84, Berlin, Heidelberg, 2010. Springer-Verlag.

[34] Yining Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2001.

[35] D. Dobkin and R. Lipton. Multidimensional search problems. *SIAM J. Computing*, 5:181–186, 1976.

[36] John P. Eakins and Margaret E. Graham. Content-based image retrieval: A report to the jisc technology applications programme. Technical report, University of Northumbria at newcastle, Institute for Image Data Research, 1999.

[37] Daniel P. W. Ellis and Keansub Lee. Features for segmenting and classifying long-duration recordings of personal audio. In *In Workshop on Statistical and Perceptual Audio Processing*, 2004.

[38] Daniel P.W. Ellis and Keansub Lee. Minimal-impact audio-based personal archives. In *Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences*, CARPE'04, pages 39–47, New York, NY, USA, 2004. ACM.

[39] M. Galassi et al. Gnu scientific library.

[40] Facebook Inc. *Facebook twitter message*. `http://techcrunch.com/2011/01/03/facebook-users-uploaded-a-record-750-million-photos-over-new-years/`.

[41] J. Fierrez-aguilar, J. Ortega-garcia, D. Garcia-romero, and J. Gonzalez-rodriguez. A comparative evaluation of fusion strategies for multimodal biometric verification. In *In Springer LNCS-2688, 4th l. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2003*, pages 830–837, 2003.

[42] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *Computer*, 28:23–32, September 1995.

[43] Imola K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[44] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. pages 296–301, 1995.

[45] Shih fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. Videoq: An automated content based video search system using visual cues. In *In Proceedings of ACM Multimedia*, pages 313–324, 1997.

[46] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. DARPA TIMIT acoustic-phonetic continuous speech corpus, 1993.

[47] P. Geetha and Vasumathi Narayanan. A survey of content-based video retrieval. *Journal of Computer Science*, 4, 2008.

[48] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221 –228, 29 2009-oct. 2 2009.

[49] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. Mylifebits: Fulfilling the memex vision, 2002.

[50] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of the 25th Int. Conf. on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

[51] Google Inc. *Google Goggles: Use pictures to search the web.* `http://www.google.com/mobile/goggles`.

[52] Anjan Goswami, Naren Chittar, and Chung H. Sung. A study on the impact of product images on user clicks for online shopping. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 45–46, New York, NY, USA, 2011. ACM.

[53] Liang Gou, Hung-Hsuan Chen, Jung-Hyun Kim, Xiaolong (Luke) Zhang, and C. Lee Giles. Sndocrank: a social network-based video search ranking framework. In *Proceedings of the international conference on Multimedia information retrieval*, MIR '10, pages 367–376, New York, NY, USA, 2010. ACM.

[54] Paul Graham. A plan for spam, august 2002. `http://www.paulgraham.com/spam.html`.

[55] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 47–57, 1984.

[56] Xian-Sheng Hua, Lie Lu, and Hong-Jiang Zhang. Optimization-based automated home video editing system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(5):572 – 583, may 2004.

[57] IDC Inc. *The 2011 Digital Universe Study*. `http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm`.

[58] Idee Inc. *Piximilar Visual Search*. `http://www.ideeinc.com/products/piximilar/`.

[59] Incogna Inc. *Incogna Visual Search Engine*. `http://www.incogna.com/`.

[60] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of the 30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[61] Iron Port Inc. *Image Spam: The E-Mail Epidemic of 2006*. `http://ironport.com/pdf/ironport_image_spam_datasheet.pdf`.

[62] Yushi Jing and Shumeet Baluja. Pagerank for product image search. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 307–316, New York, NY, USA, 2008. ACM.

[63] Alexis Joly and Olivier Buisson. A posteriori multi-probe locality sensitive hashing. In *Proceeding of the 16th ACM international conference on Multimedia*, MM '08, pages 209–218, New York, NY, USA, 2008. ACM.

[64] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Gaussian processes for object categorization. *International Journal of Computer Vision*, 88(2):169–188, 2010.

[65] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pages 369–380, 1997.

[66] Michael Kazhdan, Thomask Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proc. of the Eurographics Symposium on Geometry Processing*, 2003.

[67] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR (2)'04*, pages 506–513, 2004.

[68] Yan Ke, Rahul Sukthankar, and Larry Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, 2004.

[69] Josef Kittler, Ieee Computer Society, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:226–239, 1998.

[70] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. of the 15th ACM Symposium on Discrete Algorithms*, pages 798–807, 2004.

[71] Julien Law-To, Li Chen, Alexis Joly, Ivan Laptev, Olivier Buisson, Valerie Gouet-Brunet, Nozha Boujemaa, and Fred Stentiford. Video copy detection: a comparative study. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 371–378, New York, NY, USA, 2007. ACM.

[72] Sunil Lee and C.D. Yoo. Robust video fingerprinting for content-based video identification. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(7):983 –988, july 2008.

[73] John Levine. Experiences with greylisting. In *Second Conference on E-mail and Anti-Spam*, 2005.

[74] Xirong Li, Dong Wang, Jianmin Li, and Bo Zhang. Video search in concept subspace: a text-like paradigm. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 603–610, New York, NY, USA, 2007. ACM.

[75] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 745 – 748, aug. 2001.

[76] H. Luo, J. Fan, S. Satoh, J. Yang, and W. Ribarsky. Integrating multi-modal content analysis and hyperbolic visualization for large-scale news video retrieval and exploration. *Signal Processing: Image Communication*, 23(7):538 – 553, 2008. Special Issue on Semantic Analysis for Interactive Multimedia Services.

[77] Mathias Lux and Savvas A. Chatzichristofis. Lire: lucene image retrieval: an extensible java cbir library. In *Proceeding of the 16th ACM international conference on Multimedia*, MM '08, pages 1085–1088, New York, NY, USA, 2008. ACM.

[78] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Efficient filtering with sketches in the ferret toolkit. In *Workshop on Multimedia Information Retrieval*, 2006.

[79] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Ferret: A Toolkit for Content-Based Similarity Search of Feature-Rich Data. In *Proceedings of the ACM SIGOPS EuroSys Conf.*, 2006.

[80] Qin Lv, Moses Charikar, and Kai Li. Image similarity search with compact data structures. In *Proc. of the 13th ACM Conf. on Information and Knowledge Management*, pages 208–217, 2004.

[81] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007.

[82] Wei-Ying Ma and Hong Jiang Zhang. Benchmarking of image features for content-based retrieval. In *Signals, Systems Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, volume 1, pages 253 – 257 vol.1, nov 1998.

[83] Mail Avenger, 2006. `http://www.mailavenger.org`.

[84] S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

[85] C. W. Ngo, T. C. Pong, and H. J. Zhang. Motion analysis and segmentation through spatio-temporal slices processing. In *IEEE Trans. on Image Processing, vol. 12, no 3*, 2003.

[86] Paul Over, Tzveta Ianeva, Wessel Kraaij, and Alan F. Smeaton. Trecvid 2005 - an overview. In *In Proceedings of TRECVID 2005*, 2005.

[87] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1186–1195, Miami, Florida, USA, Jan 2006.

[88] Princeton Spam Image Benchmark. `http://www.cs.princeton.edu/cass/spam`.

[89] Calton Pu and Steve Webb. Observed trends in spam construction techniques: A case study of spam evolution. 2006.

[90] A. Ramachandran and N. Feamster. Unerstanding the nework-level behavior of spammers. *ACM SIGCOMM Computer Communication Review*, 36(4), October 2006.

[91] Riya Inc. *Riya's like.com*. `http://www.like.com/`.

[92] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13):2115 – 2125, 2003. Audio- and Video-based Biometric Person Authentication (AVBPA 2001).

[93] Y. Rubner, L. J. Guibas, and C. Tomasi. The earth movers distance, high-dimensional scaling, and color-based image retrieval. In *Proc. of the DARPA Image Understanding Workshop*, pages 661–668, 1997.

[94] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[95] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, 1998.

[96] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. The a-tree: An index structure for high-dimensional spaces using relative approximation. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 516–526, 2000.

[97] Using Large Scale and George Tzanetakis. Marsyas3d: A prototype audio browser-editor. In *In Proc. International Conference on Auditory Display*, 2001.

[98] Richard Segal, Jason Crawford, Jeff Kephart, and Barry Leiba. Spamguru: An enterprise anti-spam filtering system. In *First Conference on Email and Anti-Spam (CEAS)"*, 2004.

[99] Shazam Inc. *Shazam music*. `http://www.shazam.com/`.

[100] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[101] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-base image retrieval at the end of the early years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(12), 2000.

[102] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349 –1380, dec 2000.

[103] J.R. Smith, S. Basu, Ching-Yung Lin, M. Naphade, and B. Tseng. Interactive content-based retrieval of video. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–976 – I–979 vol.1, 2002.

[104] Robert Snelick, Umut Uludag, Alan Mink, Michael Indovina, and Anil Jain. Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:450–455, 2005.

[105] Cees G. M. Snoek and Marcel Worring. Concept-based video retrieval. *Found. Trends Inf. Retr.*, 2:215–322, April 2009.

[106] Cees G.M. Snoek and Marcel Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25:5–35, 2005. 10.1023/B:MTAP.0000046380.27575.a5.

[107] Soundhound Inc. *Soundhound*. `http://www.soundhound.com/`.

[108] SpamAssassin, 2007. `http://spamassassin.apache.org`.

[109] M. Stricker and M. Orengo. Similarity of color images. In *In SPIE Conference on Storage and Retrieval for Image and Video Databases III, volume 2420, pages 381-392,*, 1995.

[110] Symantec Inc. *The State of Spam, A Monthly Report - January 2007*. `http://www.symantec.com/avcenter/reference/Symantec_Spam_Report_-_January_2007.pdf`.

[111] Datchakorn Tancharoen, Toshihiko Yamasaki, and Kiyoharu Aizawa. Practical experience recording and indexing of life log video. In *Proceedings of the 2nd ACM workshop on Continuous archival and retrieval of personal experiences*, CARPE '05, pages 61–66, New York, NY, USA, 2005. ACM.

[112] Taobao Inc. *Taobao Network Limited*. `http://www.taobao.com/`.

[113] H. Terasawa, M. Slaney, and J. Berger. Perceptual distance in timbre space. pages 61–68, Limerick, Ireland, 2005. Department of Computer Science and

Information Systems, University of Limerick, Department of Computer Science and Information Systems, University of Limerick.

[114] M. B. Eisen *et al.* Cluster analysis and display of genome-wide expression patterns. *Proc. of National Academy of USA*, 95(25):14863–8, 1998.

[115] P. Tamayo *et al.* Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. of National Academy of Sciences of USA*, 96(6):2907–12, 1999.

[116] Tineye Inc. *Tineye Reverse Image Search*. `http://www.tineye.com/`.

[117] Antonio Torralba, Rob Fergus, and Yair Weiss. Small codes and large image databases for recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[118] Twitter Inc. *Twitter Blog*. `http://blog.twitter.com/2011/03/numbers.html`.

[119] G. Tzanetakis and P. Cook. *MARSYAS: A Framework for Audio Analysis*. Cambridge University Press, 2000.

[120] B. Vanneva. As we may think. *The Atlantic Monthly*, 176(1), 1945.

[121] Remco C. Veltkamp and Mirela Tanase. Content-base image retrieval systems: A survey. Technical Report US-CS-2000-34, Utrecht University, Information and Computer Sciences, 2000.

[122] Patrick Verlinde, Gerard Chollet, and Marc Acheroy. Multi-modal identity verification using expert fusion. *Information Fusion*, 1:17–33, 2000.

[123] Fabio Vignoli and Steffen Pauws. A music retrieval system based on user-driven similarity and its evaluation. In *In Proc. International Symposium on Music Information Retrieval*, pages 272–279, 2005.

[124] Vipul's Razor, 2007. `http://razor.sourceforge.net`.

[125] James Z. Wang, Jia Li, and Gio Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[126] Zhe Wang, Wei Dong, William Josephson, Qin Lv, Moses Charikar, and Kai Li. Sizing sketches: a rank-based analysis for similarity search. *SIGMETRICS Perform. Eval. Rev.*, 35:157–168, June 2007.

[127] Zhe Wang, William Josephson, Qin Lv, Moses Charikar, and Kai Li. Filtering image spam with near-duplicate detection. In *In Proceedings of the Fourth Conference on Email and AntiSpam*, 2007.

[128] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*, pages 194–205, 1998.

[129] Shikui Wei, Yao Zhao, Zhenfeng Zhu, and Nan Liu. Multimodal fusion for video search reranking. *IEEE Transactions on Knowledge and Data Engineering*, 22:1191–1199, 2010.

[130] Xiao-Yong Wei, Yu-Gang Jiang, and Chong-Wah Ngo. Concept-driven multi-modality fusion for video search. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(1):62 –73, jan. 2011.

[131] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. Spatio-temporal features for robust content-based video copy detection. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, MIR '08, pages 283–290, New York, NY, USA, 2008. ACM.

[132] Ching-Tung Wu, Kwang-Ting Cheng, Qiang Zhu, and Yi-Leh Wu. Using visual features for anti-spam filtering. In *IEEE International Conference on Image Processing*, volume 3, pages 509–512, 2005.

[133] Si Wu, Yu-Fei Ma, and Hong-Jiang Zhang. Video quality classification based home video segmentation. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, page 4 pp., july 2005.

[134] Xiao Wu, Alexander G. Hauptmann, and Chong-Wah Ngo. Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 218–227, New York, NY, USA, 2007. ACM.

[135] Yi Wu, Edward Y. Chang, Kevin Chen-Chuan Chang, and John R. Smith. Optimal multimodal fusion for multimedia data analysis. In *ACM Multimedia'04*, pages 572–579, 2004.

[136] Youtube Inc. *Youtube Blog.* `http://youtube-global.blogspot.com/2010/03/oops-pow-surprise24-hours-of-video-all.html`.

[137] Vojtěch Zavřel, Michal Batko, and Pavel Zezula. Visual video retrieval system using mpeg-7 descriptors. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 125–126, New York, NY, USA, 2010. ACM.

[138] Dong-Qing Zhang and Shih-Fu Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 877–884, 2004.

[139] Hong Jiang Zhang, Jianhua Wu, Di Zhong, and Stephen W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643 – 658, 1997. Image Databases.

[140] Ming Zhao, Jiajun Bu, and Chun Chen. Audio and video combined for home video abstraction. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, pages 1520–6149, 2003.