Content-Based Image Retrieval: Reading One's Mind and Helping People Share

SIA Ka Cheung

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Department of Computer Science & Engineering

September 2003

To my parents

Content-Based Image Retrieval: Reading One's Mind and Helping People Share

submitted by

SIA Ka Cheung

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Content-Based Image Retrieval (CBIR) had been proposed for nearly ten years, yet, there are still many open problems remain unsolved. The learning of image similarity, the interaction with users, the need for databases, the semantic gap with image features, and the understanding of images are keys to improve CBIR. In this thesis, we make use of relevance feedback architecture to learn image similarity through interactions with users. We model users' target as a distribution in the image feature vector space and propose a parameters estimation approach to capture it. Moreover, a Self-Organizing Map (SOM) inter-query feedback approach is used to reorganize this feature vector space so that distribution of usres' target can be captured more accurately using parameters estimation method. Experiments on both synthetic and real data show that our proposed method give a better understanding of user's target distribution and improved retrieval precision in the real data experiment.

Peer-to-Peer (P2P) systems are recently evolved paradigms for achieving higher throughput of both data storage and computation power. We foresee the potential for P2P to support CBIR in the sense of increased image collection and workload sharing of feature extraction. We developed a P2P client, called DISCOVIR, capable of performing CBIR in a P2P network. In addition, we propose a Peer Clustering and Firework Query Model to tackle the data location problem in distributed environment, like P2P. With this method, shared data and peers in a P2P network are organized like social communities, thus query routing scheme can be applied to reduce query messages broadcasting that often occurs in unstructured P2P networks. We demonstrated the scalability of our algorithm through simulation with real and synthetic data.

Acknowledgment

I would like to thank my supervisor, Prof. Irwin King, for his patient guidance, ceaseless support, insightful ideas and generous encouragement in all aspects throughout the two years of my MPhil study. I am very grateful to have him as my supervisor. He introduced me to research; polished my writing and presentation skills; gave me opportunities to attend international conferences, which benefit me for the rest of life. I am proud of being his student, now and forever.

Thanks go to Prof. Lai-Wan Chan and Prof. Yiu-Sang Moon for their invaluable comments, to Mr. Chi-Hang Chan and Mr. Cheuk-Hang Ng for sharing their knowledge and insightful discussions in my research, to team members of DISCOVIR for their efforts. In particular, I would like to thank my dear friend Janice, for her support, encouragement and understanding.

Last but not least, I would like to share some words here to respect all the scientists and engineers for their contributions which we can grounded on when developing our future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

“A picture is worth a thousand words”. What does this idiom tells us? Obviously, images play a much more important role than text. The information contained in an image even cannot be described by words. However, the search and organization of image databases is often of less important than text databases. Plainly, there must be difficulties lie ahead for us to solve.

1.1 Problem Statement

Content-Based Image Retrieval (CBIR) has been proposed for nearly ten years, yet, there are still many open problems left unsolved. According to some researchers [48, 42], the learning of image similarity; the interaction with users; the need for databases; the problem of evaluation; the semantic gap with image features; and the understanding of images are keys to improve CBIR. In our research, we target on the first three areas. Relevance Feedback (RF) [43, 3, 12, 54, 28, 19, 1, 8] is a commonly used architecture for refining queries in search engines. Under this architecture, a search engine first presents a set of images to the user; then, he marks corresponding images as relevant or non-relevant to his target. The system learns from this feedback information and presents a set of "better"[1] images to the user. The learning process is modeled either as a classification problem [10, 1, 21, 28, 53, 54] or as an estimation problem [12, 46, 7, 23, 43, 56] with feedback information being treated as labeled training data or observations respectively.

The goal of relevance feedback is to capture user's target as quickly as possible and present him a set of most relevant images to his target. Intuitively, the effectiveness and accuracy of learning algorithm is the core of improving retrieval precision. Since the relevance feedback is an interactive process, we argue that the learning phase should take the display set selection phase into consideration as the latter one imposes a prior constraint for the feedback information given by users. Thus, we aim to devise better learning methods that consider the interaction with users in order to learn more accurately the similarity of images form their feedback. We show the accuracy of our parameters estimation method using synthetic data and how it is going to improve retrieval precision in real data experiments.

Peer-to-Peer (P2P) systems/networks are recently evolved paradigms to achieve higher throughput of both data storage and computation power. As there is no central index storing data location in a typical P2P systems, the process of locating data often brings about the query message flooding problem, in which a query has to visit many peers in order to locate the source of a file. Many architectures had been proposed in the literature to address this problem. The mainstreams are to impose an overlay structure in a P2P network or to route query instead of broadcasting [51, 39, 13, 50]. However, most of them focus on the efficient discovery using a document identifier[2]. Efficient scheme for finding similar documents based on content is still needed.

We foresee the potential for P2P systems in supporting CBIR in the sense

[1] It might be better from the user's perspective (more relevant images) or from the system's perspective (a set of images to capture more information from user in the next round).

[2] It can be a filename, hash value of a filename or message digest of the file content.

of increasing images collection and workload sharing of feature extraction. Not only the data storage of CBIR can be increased drastically, but also P2P communities benefit from it for increasing the query variety by adding CBIR functionality. We outline the architecture for performing CBIR in P2P and propose a Peer Clustering and Firework Query Model to address the problem of efficient location of similar data in a P2P network.

1.2 Contributions

In this thesis, we explore the area of learning image similarity and new form of database support in CBIR. In particular,

1. We propose a parameters estimation [46] intra-query feedback approach for capturing users' searching target more accurately while computation cost is kept minimal.

2. We propose a Self-Organizing Map (SOM) -based inter-query feedback approach [7] to reorganize the feature vector space so that feature vectors of images under the same semantic meaning will be clustered like a Gaussian distribution.

3. We develop a P2P client program capable of performing CBIR called DISCOVIR [47]. It demonstrates the possibility of applying content-based similarity search in P2P instead of the filename or file ID searches in existing P2P networks.

4. We propose an overlay network architecture and query routing scheme [36, 35, 24] to reduce network traffic by eliminating query broadcasting. Although query broadcasting problem has been addressed in many new approaches, it is only solved in the context of filename or file ID searching. Our proposed strategy solves this in the context of content-based similarity searching.

1.3 Thesis Organization

In this thesis, we review current literatures on relevance feedback techniques and P2P systems in Chapter 2. In particular, three approaches [43, 12, 22] in relevance feedback and two architectures [40, 13] in P2P are visited. In Chapter 3, we propose and evaluate a parameters estimation method and a Self-Organizing Map (SOM) [25] approach used in the relevance feedback architecture. Experiments on synthetic and real data are shown to analyze the characteristics and performance of our algorithm.

In Chapter 4, we describe the implementation and incorporation with CBIR functionality of a P2P client program, called DISCOVIR. We also describe the Peer Clustering strategy to organize a P2P network and propose the Firework Query Model that reduces network traffic and improves retrieval performance. Simulation experiments are used to test the scalability of this model. Lastly, we discuss possible future works and give concluding remarks in Chapter 5.

Chapter 2

Background

Information Retrieval is the exploration or search of a database or the Internet in order to help **people** access information in a *precise* and *efficient* manner.

2.1 Content-Based Image Retrieval

As an idiom said "A picture is worth a thousand words", even a small image can convey complex structure with rich content. It seems that searching for images is much more difficult than searching for text. In early image retrieval systems, they require human annotation and classification on the image collection, the query is thus performed using text-based retrieval methods. However, there are two limitations for such implementation: **Human Intervention**[1] and **Non-Standard Description**[2]. With the rapid growth in the volume of digit images, searching and browsing in a large collection of unannotated images is gaining importance. Content-Based Image Retrieval (CBIR) systems are proposed to pass this tedious task to computer. Since early 1990's, many systems have been proposed and developed, some of them are QBIC [16], WebSEEK [49], SIMPLIcity [55], MARS [31], Photobook [38], WALRUS [34] and other systems for domain specific applications [26, 23]. In CBIR systems,

[1] The process of annotating images is potentially error prone and tedious.

[2] Different annotators may have different perception on the same image and use different words to describe it.

low-level features of images, like texture, color and shape, are first extracted as multi-dimensional vector and indexed to represent images in the retrieval process. In the following two sections, we will address two main topics in CBIR: **Feature Extraction**, and **Indexing & Retrieval**.

2.1.1 Feature Extraction

In order to search images based on their contents, a CBIR system must obtain the content-based feature of its image collection, this process is known as feature extraction. When an image is added to the collection, several features, such as color, texture and shape[3], are extracted and transformed into feature vector. Here is the mathematical model of feature extraction:

$$f : I \times \theta \rightarrow R^d \tag{2.1}$$

where f represents a specific feature extraction method, I is the image under operation, θ is a set of parameters for feature extraction f, R^d is a real value d dimension vector as the result of feature extraction. Fig. 2.1 shows the average RGB values of an image collection are extracted and represented as 3-dimensional data points.

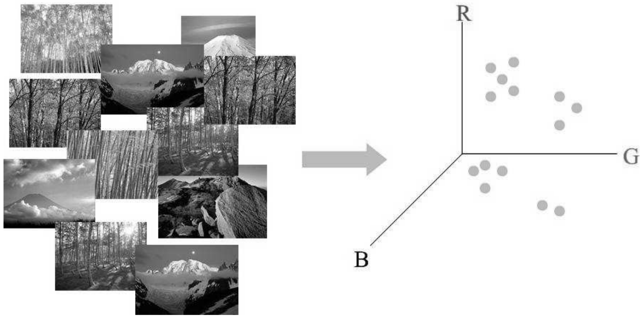


Figure 2.1: Feature extraction of average RGB value

[3] Specially, some feature likes Color Histogram, Color Moment, Color Coherence Vector, Tamura, Gabor Wavelet, Fourier Descriptor.

2.1.2 Indexing and Retrieval

After the feature vectors of images are extracted, they are used to represent the images in the retrieval process. Unlike text indexing and retrieval problems, which can be handled by B-Tree or inverted file[4][3], there are techniques developed to index and retrieve these multi-dimensional points, the most common one is R-Tree [20] and its variation [5] as shown in Fig. 2.2. The dots represent feature vectors in two dimension and bounding boxes at different level are used to organize them and build indexing structure.

In most of the CBIR systems, they assume a close distance in the low-level feature vector space to be the definition of similar images. Under this assumption, nearest-neighbor search in the indexing structure is used to locate a set of similar images in respond to a query.

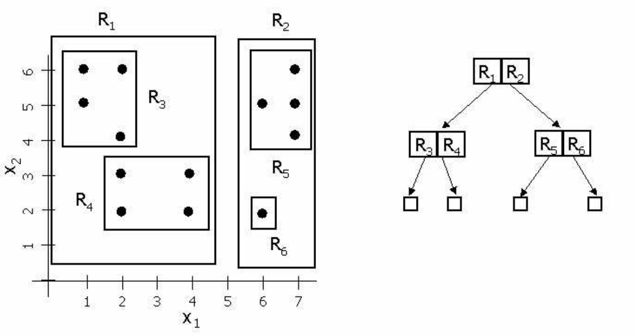


Figure 2.2: R-Tree indexing structure:(left) 2-dimensional data points and bounding rectangles, (right) indexing structure

2.2 Relevance Feedback

As described in Section 2.1, previous systems are based on the one-shot approach, in which images with the shortest distance (in the feature vector space) to the query image are retrieved and presented to the user. Plainly, this model

[4] Inverted file is a data structure commonly used to locate files/websites that contains the query term in a search engine.

is not adaptive to different users; as one might pay more attention to color feature rather than the texture feature in an image. To adapt to different users' preference, relevance feedback architecture is proposed. The goal of relevance feedback is to learn users' preference from their interaction, and it is a powerful technique to improve the retrieval results in CBIR. Under this framework, a set of images is presented to the user according to the query. The user marks those images as either relevant or non-relevant and gives feedbacks to the system. Based on this feedback, the system estimates user's preference through a learning process. Fig. 2.3 shows a typical relevance feedback process.

Most of the current relevance feedback systems are based on the intra-query approach. In this approach, the system refines the query by using feedback information supplied by the user, and this learning process starts from ground up for each query made. For example, *PicHunter* [12] presented a Bayesian framework to direct a search with relevance feedback information. Each image in the database is associated with a probability being the target image of a user's query. The Bayes's rule is used to update the probability according to user's feedback information in every iteration. Rui et al. [43] proposed a weight updating approach to capture user's preference on different feature representations of images. The weight of each feature, its representation and each dimension is updated by their discriminative power between the set of relevant and non-relevant images in the current query. The similarity measure of images is thus based on their **weighted** distance in the feature space. The intra-query approach uses the feedback information to estimate user's target distribution, but it often fails when the underlying distribution is not clustered in the low-level feature space. This is due to the feedback information given by a single query that is limited and unable to provide enough statistical information about the distribution [56]. In the following sections, we review the method proposed by Rui et al. [43], Cox et al. [12] and other researchers.

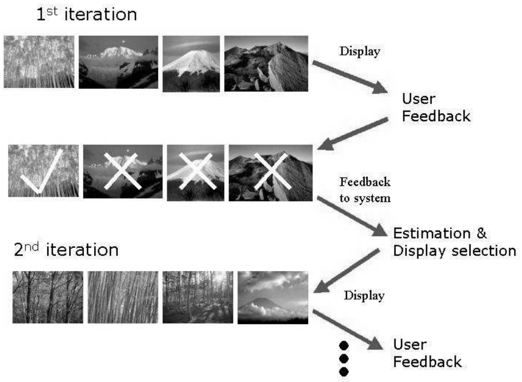


Figure 2.3: Relevance feedback architecture

2.2.1 Weight Updating

In [43], objects in image database are modeled as

$$O = O(D, F, R), \tag{2.2}$$

where D is the raw image data, $F = \{f_i\}$ is a set of low level visual features, such as color, texture, and shape, and $R = \{r_{ij}\}$ is the set of representations for f_i, which is defined as

$$r_{ij} = [r_{ij1}, ..., r_{ijk}, ..., r_{ijK}]. \tag{2.3}$$

Moreover, the feature vector is organized in a hierarchical manner. The overall similarity of two images O^a and O^b is defined as

$$\begin{aligned} S(O^a, O^b) &= \sum_i W_i S(f_i^a, f_i^b), && (2.4)\\ S(f_i^a, f_i^b) &= \sum_j W_{ij} S(r_{ij}^a, r_{ij}^b), && (2.5)\\ S(r_{ij}^a, r_{ij}^b) &= m(r_{ij}^a, r_{ij}^b, W_{ijk}), && (2.6) \end{aligned}$$

where m is the distance measure function, while W_i, W_{ij} and W_{ijk} are the weights associated with each features, its representation and each dimension

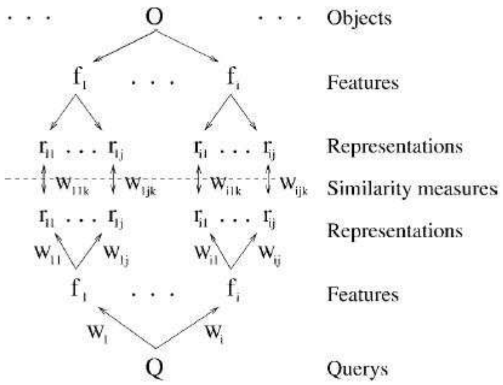


Figure 2.4: Feature organization in Rui's method

respectively. For each feedback, they will follow the two procedures described below to update the weight in order to capture user's interest in different features. Fig. 2.4 shows the organization of features, representations and dimensions in this approach.

Interweight Updating Procedure

Suppose RT is the set of most similar images presented to the user according to the overall similarity function Eq. 2.4. Under each feature representation ij, the system retrieves a set of similar images, not presenting to users, RT^{ij}, according to that particular feature representation. The weight, W_{ij}, is updated according to

$$W_{ij} = \begin{cases} W_{ij} + R, & RT_l \in RT^{ij} \text{ and } l = 1, ..., N, \\ W_{ij}, & \text{otherwise} \end{cases}$$

where N is the number of images in the set RT and R_l is the degree of relevance of image l indicated by the user. The larger the value R, the more the relevant an image is. All the W_{ij} are then normalized after this update.

Intraweight Updating Procedure

For the set of relevant images indicated in the user's feedback, the system computes the standard deviation, σ_{ijk}, in each dimension, and the weight for each dimension is updated as Eq. (2.7). We can see that if σ_{ijk} is large, then the dimension is not ideal for discriminating relevant and irrelevant images, so its weight is updated as follows. Again, all the W_{ijk} are normalized after this update.

$$W_{ijk} = \frac{1}{\sigma_{ijk}}. \tag{2.7}$$

2.2.2 Bayesian Formulation

In [12], each image is associated with a probability of being the user's target. The retrieval process consists of two steps. In each pass, the system selects a set of images and presents to user. Through the feedback, the system updates the likelihood measure to the query of each image accordingly. The probability is updated using the Bayes' rule as follows,

$$P(T = T_i|H_t) = \frac{P(A_t|T = T_i, D_t, S_{t-1})P(T = T_i|H_{t-1})}{\sum_{j=1}^{n} P(A_t|T_j, D_t, S_{t-1})P(T_j|H_{t-1})}. \tag{2.8}$$

The meaning of Eq. (2.8) is that the probability of T_i being the target image at iteration t is equal to product of the probability of T_i being the target at iteration $t-1$ and the probability of user give such feedback at iteration t provided that T_i is the target, over the summation of this product of other images. D_t is the data presented to user at iteration t, A_t is the feedback given by user at iteration t, while H_t and S_t stands for history of user's feedback at iteration t.

Moreover, as each image is associated with a probability of being the target, [12] proposed a maximum entropy display strategy to select image presenting to user. As a result, the system is expected to get most information gain from user's feedback. Besides, [23] also details the procedure to apply this strategy. This method inspires the display model of our proposal.

2.2.3 Statistical Approaches

Nigam et al. [37] and Wu et al. [56] proposed using the EM algorithm to classify documents and images respectively. Expectation Maximization (EM) algorithm was first proposed in 1977 [15] . It was used to solve the maximum-likelihood from incomplete data. Given a mixture of Gaussians, the EM algorithm estimates the parameter of each mixture, say, mean and variance. Our proposed approach is based on EM algorithm to estimate the parameter of user's target distribution which will be detailed in Chapter 3. The EM approach in [56] utilizes the information contained in unlabeled data[5] to help estimating user's target distribution when labeled data is limited. More recently, Tian et al. [53] model this as classification problem and tackle it by Support Vector Machine (SVM). By treating the marked relevant and non-relevant images (their corresponding feature vector) as support vector, SVM is used to delineate a region containing relevant images having minimized risk of incorrect classification. All these newly suggested methods tends to model the feedback process as a learning process and apply algorithms in computer learning to help.

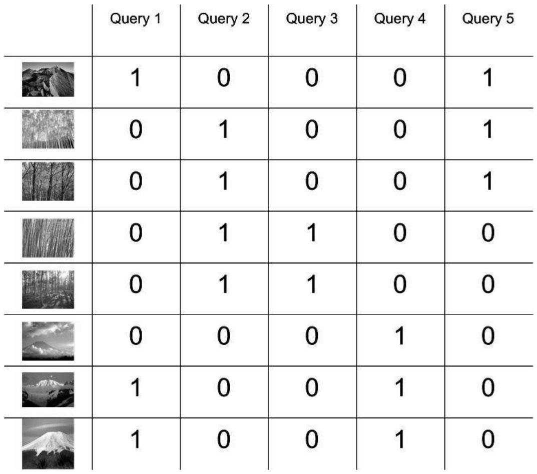
2.2.4 Inter-query Feedback

Recent researches propose the use of inter-query information to improve retrieval results. The relevance feedback not only provides information about a user's target distribution and preference, but also a similarity measure of images under human perception. In this approach, feedback information from

[5] Unlabeled data means those images not marked by the user as either relevant or non-relevant

individual users are accumulated to train the system to determine which images are of the same semantic meaning. Heisterkamp [22] and He et al. [21] use inter-query information together with latent semantic indexing (LSI) [14] in CBIR. LSI is a classical document retrieval algorithm. It analyzes the correlation of documents and terms in a collection of documents. In this approach, previous feedback information are stored in the system to build the latent semantic index. Each query result and its relevance feedback is considered as a document (a semantic meaning) and each image in the database is consider as a term/word, thus the correlations of images and different semantic meanings are revealed. However, this method can only retrieve images which had been marked at least once as relevant in the history of previous of query; thus, a certain number of previous query results must be obtained in order to make sure all images can be retrieved. Moreover, this inter-query approach does not concern about the subjectivity of different users, and the same initial query point always provide the same query result.

Figure 2.5 illustrates how the system treats images as terms and successive feedback information as documents and uses LSI method to retrieve similar images. Each column in the matrix represents a query made by one user and the set of relevant images he selected, 1 being marked as relevant and 0 otherwise. Assume there are m images in the database and k queries with relevance feedback are accumulated, the dimension of matrix R shown in Fig. 2.5 is $m \times k$, while the dot-product between row i and row j gives the similarity measure between image i and j under the semantic space learned from user. In practice, different queries may involve common high-level semantic features; thus R is decomposed using singular value decomposition to $R = UST^T$, with $U^TU = I$, $V^TV = I$ and S diagonal. Column vectors of U and V are eigenvectors of RR^T and R^TR respectively. R is then represented by a compact form (hidden semantic space) B, which $B = US$ and it requires reduced storage space $m \times p$, p being the rank of matrix R and $p < k$.



	Query 1	Query 2	Query 3	Query 4	Query 5
	1	0	0	0	1
	0	1	0	0	1
	0	1	0	0	1
	0	1	1	0	0
	0	1	1	0	0
	0	0	0	1	0
	1	0	0	1	0
	1	0	0	1	0

Figure 2.5: LSI model of inter-query feedback

Table 2.1: Comparison of different relevance feedback approaches

Methods	Pros	Cons
Weight updating	Fast	Inadequate to discover semantic meaning
Probabilistic framework	Utilize users' feedback by maximum entropy principle	High computation cost
LSI inter-query feedback	Capture semantic meaning	Unable to adapt to different users' preference
Statistical approaches	Sound theoretical background	Passive - the learning algorithm is applied using random sampling
Parameters estimation	Fast, utilized users' feedback information	Require a training phase to reorganize feature space

The following is a comparison of different relevance feedback approaches.

2.3 Peer-to-Peer Information Retrieval

Peep-to-Peer (P2P) network is a recently evolved paradigm for distributed computing. With the emerging P2P networks or their variants such as Gnutella [18], Napster [33] and Freenet [17], and their implementations, such as LimeWire [29], etc. They offer the following advantages:

1. **Distributed Resource**–The storage, information and computational

cost can be distributed among the peers, allowing many individual computers to achieve a higher throughput [45].

2. **Increased Reliability**–The P2P network increases reliability by eliminating reliance on centralized coordinators that are potential critical points of failure [11].

3. **Comprehensiveness of Information**–The P2P network has the potential to reach every computers on the Internet, while even the most comprehensive search engine can only cover one-third of the **indexable**[6] web-site available as stated in some statistics. [27]

Figure 2.6 shows an classical example of a P2P network[7]. In the example, different files are shared by different peers. When a peer initiates a search for a file, it broadcasts a query request to all its connecting peers. Its peers then propagate the request to their own peers and this process continues. Unlike the client-server architecture of the web, the P2P network aims at allowing individual computers, which join and leave the network frequently, to share information directly with each other without the help of dedicated servers. Each peer acts as a server and as a client simultaneously. In these networks, a peer can become a member of the network by establishing a connection with one or more peers in the current network. Messages are sent over multiple hops from one peer to another while each peer responds to queries for information it shares locally. Plainly, this model is wasteful because peers are forced to handle irrelevant query messages. This type of search is called a Breadth-First-Search (BFS). As shown in this example, the data location process is a key problem in P2P information retrieval. When a peer issues a query request, the P2P system must be able to locate the required data in an efficient manner. Many

[6] Indexable here means not including the non-indexable web-site, which is many times the indexable one as stated in some statistics [6].

[7] It is in fact the Gnutella model.

methods are proposed to solve this problem. They can be divided into two categories and we are going to introduce them in the following two sections.

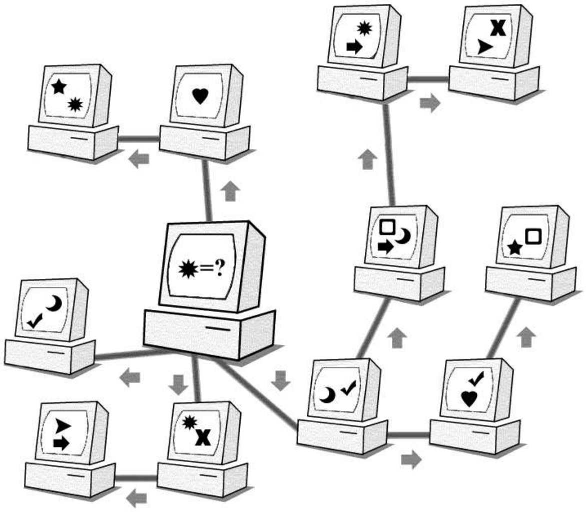


Figure 2.6: Illustration of information retrieval in a P2P network

2.3.1 Distributed Hash Table Techniques

To address the data location problem, Chord [51], CAN [39], Pastry [41] and Tapestry [59] tackle it by distributing the index storage into different peers, thus sharing the workload of a centralized index server. The distributed infrastructure uses Distributed Hash Table (DHT) technique to map the filename of files shared by a peer to keys, and each peer is responsible for storing a certain range of (Key, Value)[8] pairs. When a peer looks for a file, it hashes the filename to a key and asks the peers responsible for this key for the actual storage location of that file. Chord models the key as an m-bits identifier and arranges the peers into a logical ring topology to determine which peer is responsible for storing which (Key, Value) pair. CAN models the key as point on a d-dimension Cartesian coordinate space, while each peer is responsible

[8] Key is the hashed value of filename and Value is the filename and location.

for (Key, Value) pairs inside its specific region. Fig. 2.7 illustrates how CAN distributes indices into different peers. Such systems take a balance between the centralized index and totally decentralized index approaches. They speed up and reduce message passing for the process of key lookup (data location). Some extensions of DHTs to perform content-based retrieval and textual similarity matches instead of filename match are proposed in [52]. Although DHTs are elegant and scalable, their performance under the dynamic conditions for P2P systems is still unknown[40]. Moreover, such kind of schemes rely on the trustworthiness of peers participating in the network. If a malicious peer which supposed to be responsible for answering queries deny to do so, the problem becomes serious, especially under the condition of no duplicate index storage in other peers.

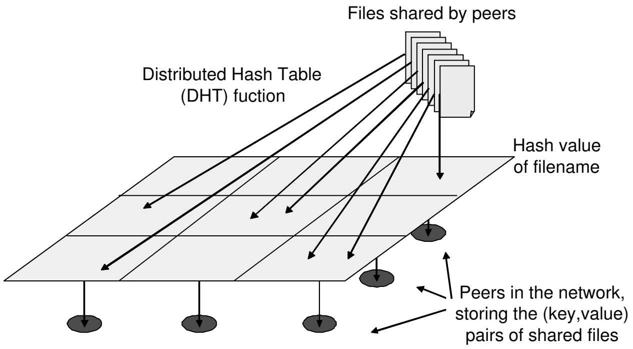

Files shared by peers
Distributed Hash Table (DHT) fuction
Hash value of filename
Peers in the network, storing the (key,value) pairs of shared files

Figure 2.7: Illustration of indices management in CAN

2.3.2 Routing Indices and Shortcuts

As DHTs mandate a specific network structure and queries are based on document identifiers, researchers proposed methods that operate under the prevalent dynamic P2P environment, like Gnutella, and queries are based on content of documents. Crespo [13] proposed a routing indices approach for retrieving text documents in P2P systems. Under this scheme, each peer maintains a routing index to assist in forwarding queries to peers that contain more documents of the same category as the query. Fig 2.8 illustrates how the routing index is created and propagated to neighboring peers. DB, N, T and L represent databases, networks, theory, and languages categories respectively. Each row in the routing index is the "estimated" number of documents under different categories shared by the corresponding peer. Figure 2.8 (a) shows how A and D exchange their routing index while (b) shows how the index is being propagated to neighboring peers. This method requires all peers to agree upon a set of document categories and regular updates of routing indices among peers. Sripanidkulchai et al. [50] proposed the use of "short-cuts" to connect a peer to another one which it has previously downloaded documents from. The intuition behind is that a peer should probably be interested in other files of a peer which it has downloaded files from previously. Evaluations are done based on text document retrieval and promising results are shown. Our proposed method targets on CBIR in P2P network. It makes use of strategies similar to routing indices and short-cuts to reduce network traffic and computation time. Similar problems of CBIR in distributed databases have been described in [9]; however, more research are need in the P2P systems.

2.3.3 Content-Based Retrieval in P2P Systems

Most of the existing P2P applications and recently proposed DHT lookup services focus on the retrieval of documents based on the matching of filename or message digest[9], a.k.a. fingerprint, of file content. All these methods target on the exact match of a file. How about similar match? What if I want to find documents introducing XML? What if I want to find images similar to an image given by me? These questions cannot be answered by existing methods; thus, we foresee there is a pressing need for P2P document sharing systems to

[9] Common ones are SHA and MD5.

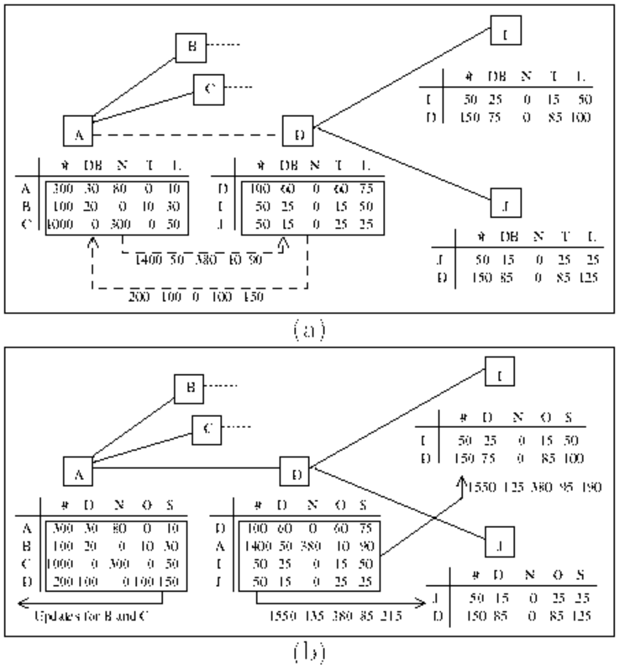


Figure 2.8: Creation and propagation of routing index

support content-based retrieval. Yang extended the MACSIS [57] framework for content-based music indexing and retrieval into a P2P environment [58] for the sake of utilizing peers' computation power in extracting characteristic sequence of shared collection. A recently proposed query routing strategy by Crespo et al. [13] also focus on the content-based documents retrieval problems in P2P network. Tang et al. [52] discuss the possibility of applying content-based retrieval under the DHT framework. All the above inspire us to design and add CBIR functionality into an existing P2P system. Chapter 4 will have an in-depth discussion about this.

Table 2.2 shows a brief comparison of the "extincted" Napster, Gnutella, DHT-based architectures and DISCOVIR on several key properties of P2P network.

Table 2.2: Comparison of different P2P architectures

Properties	**Napster**	**Gnutella**	**DHT**	**DISCOVIR**
Index storage	Centralized	Distributed	Grouped& distributed	Distributed
File storage	Distributed among peers			
Message passing	between peers and servers	among peers	among peers and routed	
Potential critical point of failure	Index server	Nil		
Lookup method	Partial filename and metadata match		Exact filename and its hash value	Feature content of images
Restrictions on overlay topology	No		Yes	
Small-world graph? [a]	No	Yes	No	Yes

[a] It tests whether the branching degrees of nodes follow a power-law distribution.

Chapter 3

Parameter Estimation-Based Relevance Feedback

How are we going to estimate a distribution correctly if random sampling is not allowed? Under the relevance feedback framework, we model the user's target as a distribution in the feature space. Our goal is to estimate this distribution based on limited sampling points because only a limited number of feedbacks are to be given by user[1]. This motivates us to tightly integrate two phases in relevance feedback in order to **actively** choose samples when estimating the distribution.

3.1 Parameter Estimation of Target Distribution

3.1.1 Motivation

An intra-query relevance feedback process is divided into two phases. The first phase is learning from users' feedback to understand his searching target. The second phase is selecting a set of images to display. The second one serves for

[1] It is nonsense to present the user a large set of images in order to take random samples.

two purposes, one is to retrieve the set of images most likely to be users' target, another one is to retrieve the set of images that helps understanding users' target most. In the following, we describe the motivation of our strategies in these two phases, then follow by detailed procedures in later sections.

Parameter Estimation as Learning

In [43], the weight updating method is a distance based similarity measure. The weight is a measure of how important a particular feature or a dimension is, in the query process. It makes use of the weighted distance in calculating the similarity. While in [12], a global update of probability, using Bayes' Rule, to all images in database after each feedback iteration is needed. It is not parametric based, and the global updating process seems to be the computational bound when the size of image database grows. In our proposal, we assume the user's target distribution follows a Gaussian and estimate the parameters of it. With these parameters, we can capture user's need more accurately compared to assigning weight to different dimensions, and the process of selecting images to display becomes easier.

Most Information Display

Many other approaches always overlook the process of selecting images to display. If we keep displaying the most similar images to user, we have no way to capture users' need in a broader sense. In [12], the most informative display set selection scheme tries to achieve this, since each image is associated with a probability value, the maximum entropy principle is applied. A sub-set of images having the highest entropy value is selected. However, when the size of image database is large, the number of permutations is huge, so a sampling approach is used to choose images that maximize the entropy. In our proposal, as we have estimated the parameters of user's target distribution, thus we propose to select images located near the **boundary** of target distribution, as illustrated in Fig. 3.1. This is analogous to the maximum entropy selection. Through this way, we can have most information gained from users' feedback. Moreover, as the display set is purposely selected to be most informative, the estimation strategy should fit this model accordingly. Instead of simply calculating the variance of relevant images as an estimate of the parameters, we use a modified approach which will be detailed in Section 3.1.3.

3.1.2 Model

We use a statistical learning method, expectation maximization (EM) to estimate the distribution of user's target of search, together with a display set selection method that fully utilize the information embedded in users' feedback in classifying relevant and irrelevant images.

Let $DB = \{I_i\}_{i=1}^{n}$ be a set of database objects, and a set of feature parameters be $\theta = \{\theta_i\}_{i=1}^{m}$. For each image in the database, we perform low level feature extraction to map it to a high dimensional data point by function f, which extracts a real-valued d-dimensional vector as,

$$f : I \times \theta \to R^d, \tag{3.1}$$

where θ_i means a specific feature, for example, the color histogram, the co-occurrence matrix based texture feature or the Fourier descriptor. Then an image will be mapped to a high dimensional vector, R^d, of dimension d. We assume the user's searching target distribution forms a cluster in the high dimensional space. Our goal is to estimate this distribution as accurately as possible, based on the user's feedback. We focus on one feature at this moment first. For each dimension under this feature, we estimate the mean, μ and variance, δ^2 of the user's target distribution.

3.1.3 Relevance Feedback

Resolving conflicts

In each pass, the system presents the user a set of images. The user then indicates whether these images are relevant or not. Let R^+ and R^- be the set of relevant images and non-relevant images in each pass respectively. Let $Rel(I_i)$ be the relevance measure of an image I_i and be defined as,

$$Rel_{t+1}(I_i) = Rel_t(I_i) + 1 \quad , \quad I_i \in R^+ \tag{3.2}$$

$$Rel_{t+1}(I_i) = Rel_t(I_i) - 1 \quad , \quad I_i \in R^-. \tag{3.3}$$

We only consider images of $Rel(.) > 0$ and $Rel(.) < 0$. The images are divided into two classes, the relevant one and the irrelevant one. We label the two sets as I^+ and I^- respectively. Using equation Eq. (3.2) and Eq. (3.3), we resolve the conflict between successive feedbacks given by user when he marks an image as relevant in one iteration while non-relevant in another iteration.

Parameter Estimation from Feedback Information

Upon at least three relevant images are indicated by the user, the system starts to estimate the mean and variance of user's target distribution in each dimension by the EM algorithm. In Fig. 3.1, we try to fit a Gaussian distribution having the data points (of relevant images) selected by user to be located in the boundary region; in other words, we are going to maximize the expression in Eq. 3.4, which is shown in Fig. 3.2.

$$E = \sum_{I_i \in I^+} \sum_{j=1}^{d} P(I_{ij}|\theta_j) \times (\frac{1}{\sqrt{2\pi}\delta_j} - P(I_{ij}|\theta j)) \tag{3.4}$$

$$P(I_{ij}|\theta j) = \frac{1}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} \tag{3.5}$$

where j is the index for dimension (from 1 to d) and θ_j denotes the combination of μ_j and δ_j for a particular dimension j. I^+ is the set of relevant images and I_{ij} is the value of feature vector of image I_i in dimension j. The reason why we use this expression is that most of the images we give to user to distinguish will fall in the area of **medium** likelihood (boundary case), so we fit our maximum likelihood function in order to make those relevant images marked by user to be appear in the **medium** likelihood region. In particular, the covariance matrix is not used and we assume independence between different dimensions because the estimation of covariance matrix is inaccurate if only few relevance feedback are given. In case number of relevance feedback given < dimension of feature vector, we are unable to update the covariance matrix.

For updating the mean, it is trivial to find the average of all relevant data, i.e.,

$$\mu_j = \frac{\sum_{I_i \in I^+} I_{ij}}{|I^+|}. \tag{3.6}$$

In order to find the best fitting δ_j, we differentiate E with respect to δ_j and assuming independency of each dimension, we get the following expression.

$$\frac{dE}{d\delta_j} = 0 \tag{3.7}$$

$$\sum_{I_i \in I^+} \Big(\frac{-1}{\pi\delta_j^3} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} + \frac{(I_{ij}-\mu_j)^2}{2\pi\delta_j^5} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}}$$

$$+ \frac{1}{\pi\delta_j^3} \exp^{-\frac{(I_{ij}-\mu_j)^2}{\delta_j^2}} - \frac{(I_{ij}-\mu_j)^2}{\pi\delta_j^5} \exp^{-\frac{(I_{ij}-\mu_j)^2}{\delta_j^2}} \Big) = 0 \tag{3.8}$$

As it is hard to make δ_j on one side in Eq. 3.8. We make use of the parameter calculated from the previous step, θ_{old_j} to derived the new parameter, θ_j. We substitute $\frac{\delta_j^{\frac{1}{2}}}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}}$ by $\delta_{old_j}^{\frac{1}{2}} P(I_{ij}|\theta_{old_j})$, and come up with an update equation for δ_{new_j}, detailed derivation is listed in Appendix A.

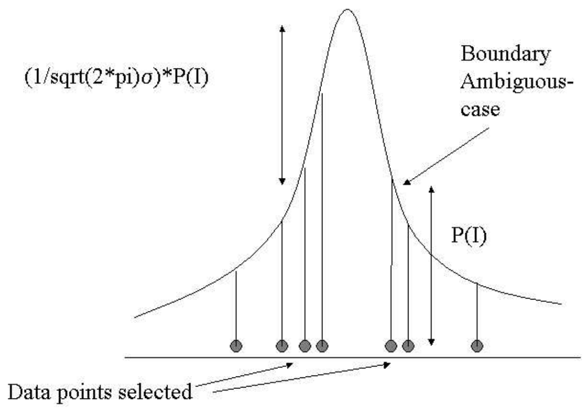


Figure 3.1: Fitting μ and σ for data points selected to display

$$\delta^2_{new_j} = \frac{\sum_{I_i \in I^+}((I_{ij} - \mu_{old_j})^2 2^{\frac{1}{4}} \pi^{-\frac{3}{4}} - \delta^{\frac{1}{2}}_{old_j} P^{\frac{1}{2}}_{old} (I_{ij} - \mu_{old_j})^2 2^{\frac{1}{2}} \pi^{-\frac{1}{2}})}{\sum_{I_i \in I^+}(2^{\frac{1}{4}} \pi^{-\frac{3}{4}} - \delta^{\frac{1}{2}}_{old_j} P^{\frac{1}{2}}_{old} 2^{\frac{1}{2}} \pi^{-1})} \tag{3.9}$$

3.1.4 Maximum Entropy Display

Since we have captured the μ and δ of user's target distribution, we proceed to select images that lie on the boundary of this distribution to display. These images are then presented to users to determine whether they are relevant or non-relevant. We choose images that located at $\pm k\delta$ away from the μ such that $P(\mu \pm k\delta) = \frac{1}{\sqrt{2\pi}\delta} - P(\mu \pm k\delta)$. Obviously, $k = 1.1774$ is the solution to this equation. In our experiment, the system selects points around $\mu + k\delta$ or $\mu - k\delta$ in each dimension to be displayed. This is analogous to the maximum entropy display as we choose the images having half probability as the most probably target image; thus, we fully utilize the power of human judgement in distinguishing different image classes. An experiment is used to verify this display select strategy and will be discussed in Section 3.3.1.

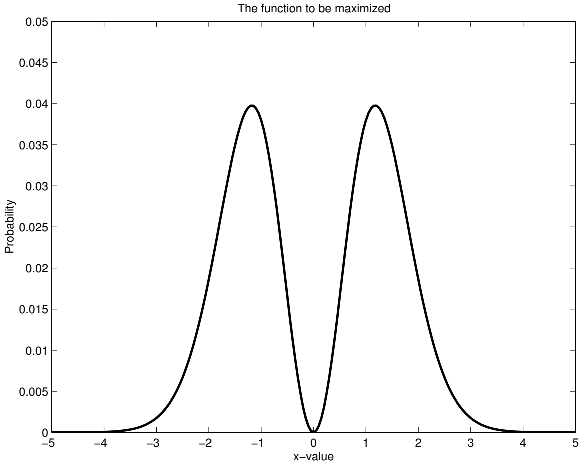


Figure 3.2: The expectation function to be maximized

Several ad-hoc based subsitutions for this maximum entropy display are available, like the K-NN method plus some random images selected from the database, or the retrieval of images from K to 2K best match. Both of them make use of including images not likely to be the estimated user's target in order to understand the actual target more. However, both of them lack a strong theoretical framework to work out the detail implementation, like the number of random images to be included, the number of k to achieve optimal information gain from user's feedback.

3.2 Self-Organizing Map Based Inter-Query Feedback

3.2.1 Motivation

In various relevance feedback systems, only the intra-query feedback information is used to estimate the user's target distribution. However, a small training data set is difficult to provide enough statistical information for estimating the underlying distribution and providing good retrieval result [56]. If the form of underlying density is known, the parameters of the density can be estimated classically by maximum likelihood. However, the underlying distribution is often not properly clustered and difficult to assume it to follow any particular form of distribution. Hence, most of the current inter-query relevance feedback systems use feedback information by different users to refine or assist in modifying the similarity measure between images only. However, these systems [22, 21] often introduce additional feature space, also known as semantic feature, that further lengthen the dimension of feature vector. Some of them even only display images that have previously been marked by users as relevant because semantic information can only be captured when they are marked by users at least one time.

In order to address the above difficulties, we use the inter-query information to modify the feature vector space and organize the neurons into Gaussian-like distributions. Thus, an prior form of density can be assumed for user's target distribution in the modified feature vector space while eliminating the need of introducing a new semantic feature space. The key idea in inter-query feedback is that a user's feedback is not only providing information for optimizing his own query, but also similarity measure between images in the database in the sense of human perception. In the proposed approach, we update the similarity measure between images dynamically according to the feedback information given by each query. It is achieved by **further** training the neurons on the SOM. Neurons represent relevant images are moved closer to the estimated user target and those represent non-relevant images are moved away from the estimated user target.

Figure 3.3 shows a two-dimensional feature vector space of a collection of images with 4 different classes. A SOM is trained based on the underlying distribution. In analyzing the image data, images from the same class often form clusters which are sparse and irregular in shape. This makes the retrieval process more difficult to find target images. With the help of inter-query feedback information described above, we organize the feature vector space in a fashion that ease the retrieval process.

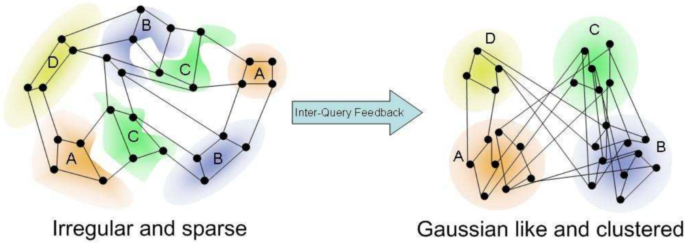


Figure 3.3: Inter-query feedback illustration

In this section, we describe our approach that utilizes both inter- and intra-query feedback information for modifying the feature vector space and estimating user's target distribution simultaneously. Self-Organizing Map (SOM) [25] is used to cluster and index images in the database; then, inter-query feedback information are used to modify the feature vector space indirectly through the SOM constructed. This allows for transforming the images distributions and improving the similarity measure. Moreover, We present an parameter estimation[2] approach to estimate user's target distribution along successive feedback given by users on the modified feature vector space. Section 3.2 introduce the construction and replication of SOM; then, the updating of this SOM by different users' feedback is detailed in Section 3.2.3. Lastly, the parameter estimation and display set selection of intra-query will be revisited in Section 3.2.4.

3.2.2 Initialization and Replication of SOM

We first perform a low-level feature extraction on the set of images $\{I_i | 1 \leq i \leq n\}$ in the database, and each image is then represented by a feature vector in a high dimensional vector space as stated in Section 3.1.2. We construct a SOM M based on the set of extracted feature vectors. After this training

[2] The original version was described in Section 3.1 and [46].

phase, the model vectors $\vec{m} \in M$ of neurons of M are used stored to partition the feature space; consequently, each image I_i is grouped into different neurons according to a minimum Euclidean distance classifier. By doing so, the size of data is reduced from $|I|$ to $|M|$, where $|I|$ and $|M|$ are the number of images and neurons respectively. In other words, the SOM M is viewed as a higher level data abstraction of underlying distribution.

The correspondence between neurons and images in the database depends on the coordinates of model vectors, any changes in the model vectors of neurons may alter this relationship. Since our proposed approach modifies the model vectors in the SOM to capture inter-query feedback information; thus, we duplicate another SOM from the original one. This new SOM contains a set of neurons with model vectors $\vec{m}' \in M'$, and has a one-to-one mapping, $f : M \to M'$, between the set M and M'. To obtain the set of images represented by model vector $\vec{m}'$, we can get the original model vector $\vec{m}$ by f^{-1}, and then by a lookup table established in the construction of M described in previous paragraph. Fig. 3.5 illustrates the retrieval process through the two SOM M and M'.

Initially, the layout of the two SOMs are the same, that is

$$\begin{aligned} &\forall \vec{m}' \in M', \forall \vec{m} \in M \\ &\qquad \vec{m}' = f(\vec{m}) \quad \Rightarrow \quad \vec{m}' = \vec{m}. \end{aligned} \tag{3.10}$$

As we alter the similarity measure between image feature vector by modifying model vectors in M' instead of M (the update method will be described in next section), the correspondence between images in the database and model vectors in M can be preserved during the whole learning process.

3.2.3 SOM Training for Inter-query Feedback

To update the similarity measure from the inter-query feedback information, we modify model vectors $\vec{m}'_i$ in the new SOM M', such that neurons contain similar images as indicated in users' feedback are moved closer to each others. In the inter-query feedback learning, we consider each query by an user as an iteration in the learning process. Assume in the k^{th} iteration, an user marked a set of relevant images I_R and a set of non-relevant images I_N during his **whole** retrieval process, M'_R and M'_N are the sets of corresponding neurons respectively. Let $\vec{c}$ be the vector having the largest probability to be the user's target in the vector space of M', and it is defined by

$$\vec{c} = \arg\max_{\vec{v}} P(\vec{v}|\theta), \tag{3.11}$$

where θ is the estimated parameters of target distribution and $P(\vec{v}|\theta)$ is the probability function of $\vec{v}$ to be the user's target and it is described in section 3.2.4. We modify the model vectors with the following equations,

$$\begin{aligned} &\forall \vec{m}'_r \in M'_r && \\ &\vec{m}'_r(k+1) &=& \quad \vec{m}'_r(k) + \alpha_R(k)(\vec{c} - \vec{m}'_r(k)), \qquad (3.12) \\ &\forall \vec{m}'_n \in M'_N && \\ &\vec{m}'_n(k+1) &=& \quad \vec{m}'_n(k) + \alpha_N(k)(\vec{m}'_n(k) - \vec{c}), \qquad (3.13) \end{aligned}$$

where $\alpha_R(k)$ and $\alpha_N(k)$ are the learning rates at k^{th} iteration and they are monotonic decreasing functions. Thus, neurons represent relevant images are moved closer to the estimated user's target and those represent non-relevant images are moved away from the estimated user's target. For a long run, the vector space will be modified, in which neurons representing different image classes are organized as separated Gaussian-like clusters on the modified feature vector space as shown in Fig. 3.3.

In a SOM, the nearby neurons in the topology are representing similar units, so that the learning process can be improved by moving also the neurons that near to the neurons in the sets M'_R and M'_N. Thus, the equations for modifying the model vectors are defined by,

$$\begin{array}{rcl} \forall \vec{m}'_h \in H(\vec{m}'_r) & , & \forall \vec{m}'_r \in M'_R \\ \vec{m}'_h(k+1) & = & \vec{m}'_h(k) + \alpha_R(k)\Lambda(\vec{m}'_h, \vec{m}'_r)[\vec{c} - \vec{m}'_h(k)], \end{array} \qquad (3.14)$$

$$\begin{array}{rcl} \forall \vec{m}'_h \in N(\vec{m}'_n) & , & \forall \vec{m}'_n \in M'_N \\ \vec{m}'_h(k+1) & = & \vec{m}'_h(k) + \alpha_N(k)\Lambda(\vec{m}'_h, \vec{m}'_n)[\vec{m}'_h(k) - \vec{c}], \end{array} \qquad (3.15)$$

where $H(\vec{m}')$ is the set of neighboring neurons for $\vec{m}'$, Λ is the neighborhood function. The most commonly used is a Gaussian neighborhood function and is defined as

$$\Lambda(\vec{m}'_h, \vec{m}'_r) = \exp\Big(-\frac{\rho(\vec{m}'_h, \vec{m}'_r)}{2\sigma^2}\Big), \qquad (3.16)$$

where σ is the spread of this Gaussian neighborhood function, and $\rho(\vec{m}'_h, \vec{m}'_r)$ is the distance between neurons $\vec{m}'_h$ and $\vec{m}'_r$ on the SOM M' grid. An illustration of this neighborhood function Λ is shown in Fig 3.4.

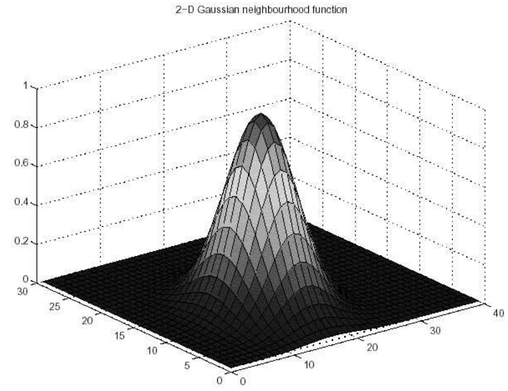


Figure 3.4: Gaussian neighborhood function on a SOM grid

3.2.4 Target Estimation and Display Set Selection for Intra-query Feedback

In the intra-query learning process, the system presents a set of images I_t to the user in each iteration (**not** the iteration in inter-query feedback), and the user marks them as either relevant or non-relevant. The system uses the set of relevant images I_{Rt} and the set of non-relevant images I_{Nt} to refine the query. After multiple iterations of inter-query feedback, the distribution of similar neurons[3] in the vector space of M' are more or less follow Gaussian distribution. We perform the user's target estimation on this vector space instead of the original feature vector space. We define M'_R as the set of relevant model vectors in M', and we use the parameter estimation based approach proposed in [46] and Section 3.1 to estimate user's target distribution. The following is basically a rewrite of Eq.3.4 in Section 3.1.3, with image feature vector I_{ij} changed to model vector of neuron m'_{ij} since we are working on the modified neurons' model vector space for the sake of reduced data-size and improved similarity measure.

$$E = \sum_{\vec{m}'_r \in M'_R} \sum_{j=1}^{d} P(m'_{rj}|\mu_j, \delta_j) \times (\frac{1}{\sqrt{2\pi}\delta_j} - P(m'_{rj}|\mu_j, \delta_j)), \tag{3.17}$$

$$P(m'_{rj}|\mu_j, \delta_j) = \frac{1}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(m_{rj}-\mu_j)^2}{2\delta_j^2}}, \tag{3.18}$$

E is the expression we want to maximize, and j is the index for dimension, ranging from 1 to d. $P(.)$ is a Gaussian density function for a model vector to be the user's target in a particular dimension. By differentiate the expression E *w.r.t.* μ_j and δ_j, the parameters update equation is as follows,

[3] Similar neurons means their corresponding images are marked as relevant to the query image by the user.

$$\mu_{new_j} = \frac{\sum_{\vec{m}'_r \in M'_R} m'_{rj}}{|M'_R|}, \tag{3.19}$$

$$\delta_j^2 = \frac{\sum_{\vec{m}'_r \in M'_R}((m'_{rj} - \mu_{old_j})^2 2^{\frac{1}{4}}\pi^{-\frac{3}{4}} - \delta_{old_j}^{\frac{1}{2}} P_{old}^{\frac{1}{2}}(m'_{rj} - \mu_{old_j})^2 2^{\frac{1}{2}}\pi^{-\frac{1}{2}})}{\sum_{\vec{m}'_r \in M'_R}(2^{\frac{1}{4}}\pi^{-\frac{3}{4}} - \delta_{old_j}^{\frac{1}{2}} P_{old}^{\frac{1}{2}} 2^{\frac{1}{2}}\pi^{-1})}. \tag{3.20}$$

In order to utilize the user's feedback information, we choose the images lie on the boundary of estimated target distribution to present to the user. This images selection strategy is analogous to the maximum entropy display as stated in [23]. Specifically, we choose the model vectors that are $\pm k\delta$ away from the μ , detail of which is stated in Section 3.1.4, After model vectors in M' are obtained, we use the function f^{-1} to obtain the neurons in the original SOM M and its corresponding images. Thus, images which are best for identifying user's target distribution are selected. The retrieval process is shown in Fig. 3.5.

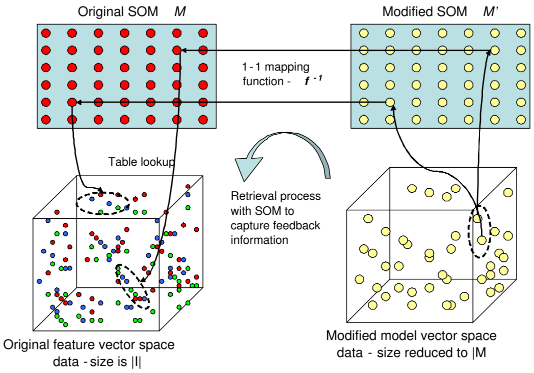

Original SOM M
Modified SOM M'
1-1 mapping function - f -1
Table lookup
Retrieval process with SOM to capture feedback information
Original feature vector space data - size is |I|
Modified model vector space data - size reduced to |M

Figure 3.5: Retrieval process using modified SOM M'

3.3 Experiment

In this section, we describe experiments, using both synthetic and real data, that verify the correctness of our parameter estimation approach and evaluate the performance of retrieval accuracy after acquiring intra- and inter-query feedback information. Section 3.3.1 describes experiments studying the convergence and performance of parameter estimation using synthetic data. Section 3.3.2 studies the improvement on retrieval accuracy of our proposed SOM-based inter-query feedback using real data.

3.3.1 Study of Parameter Estimation Method Using Synthetic Data

We setup a set of experiments to verify the correctness and measure the performance of our proposed algorithm in Section 3.1. We want to ensure the convergence property of parameter estimation is met. Moreover, we compare our proposed method with the Rui's method [43] in terms of retrieval accuracy. Here is the experimental procedure.

1. We synthetically generate a mixture of Gaussians with class labels to model different classes of image.

2. Base on our proposed algorithm, the program selects 18 data points in each iteration, and presents their class label to user.

3. The user choose one class number as his target, if he find data points come from his target, he gives feedback to program indicating that these data points are relevant.

4. After several iteration, we see if the estimated parameter of the Gaussian distribution converge towards the parameters used in the generate phase.

Table 3.1: Parameters of the synthetic dataset

Dimensionality	**Number of classes**	**Number of data points per class**	**Range of μ**	**Range of δ**
4	50	50	[-1,1]	[0.2,0.6]
6	70	50	[-1.5,1.5]	[0.2,0.6]
8	85	50	[-1.5,1.5]	[0.15,0.45]

5. We use Root-Mean-Square error to measure the difference between actual and estimated μ and δ.

Experimental Setup

In the experiments we focus only on synthetic data sets. These data sets are generated by Matlab as mixture of Gaussians. We specify the mean and variance for each class and use Matlab random function to generate. Our experiments were performed on program written in C++ running on Sun Ultra 5/400 with 256Mb ram. The parameters used to generate this synthetic dataset are listed in Table 3.1. The mean, μ, and the standard deviation, δ, of each class distributed uniformly within the range stated respectively.

The motivation of this experiment is to verify our proposed estimation method that it is able to converge to actual parameters of distribution with limited number of relevance feedback information supplied. The further elaboration of performance under realistic images are detailed in section 3.3.2.

Convergence Study

To test the convergence property, we measure the Root-Mean-Square (RMS) error between the estimated parameters and actual parameters along each iteration. Table 3.2-3.4, Fig. 3.6 show the RMS error of estimated mean and standard deviation along each iteration. The fields indicated as **not applicable** are those with fewer than 3 relevant samples given. It is because our algorithm starts to estimate the mean and standard deviation when at least 2 and 3 relevant data points are accumulated respectively. The data below

Table 3.2: Four dimensional test case results

Iteration	**Feedback Given**	**RMS mean**	**RMS std**
1	1.5	not applicable	not applicable
2	1.5	0.292545	0.20655
3	5.5	0.217373	0.203525
4	6.5	0.19565	0.180268
5	5.75	0.202975	0.16099
6	9.25	0.156245	0.134668
7	7	0.154993	0.1253
8	5.25	0.146323	0.116223
9	7	0.13309	0.111628

Table 3.3: Six dimensional test case results

Iteration	**Feedback Given**	**RMS mean**	**RMS std**
1	1.25	not applicable	not applicable
2	4	0.269095	not applicable
3	3.75	0.237395	not applicable
4	4.25	0.23813	0.182255
5	2.75	0.286803	0.172855
6	7.25	0.207565	0.136693
7	9.5	0.1705	0.122663
8	8.5	0.151863	0.122808
9	9.5	0.155308	0.121773
10	8.25	0.143003	0.10449

for each dimension is an average of 4 test cases in that particular dimension data-set.

Performance study

We have implemented the intraweight updating version (Eq. 2.7) of Rui's approach to compare with our proposed approach to study the improvement. This experiment uses the same set of synthetic data , which is a mixture of

Table 3.4: Eight dimensional test case results

Iteration	**Feedback Given**	**RMS mean**	**RMS std**
1	1.75	0.203065	not applicable
2	5.25	0.22882	0.13232
3	9.25	0.215707	0.087263
4	9.75	0.176893	0.059613
5	12	0.215953	0.059937
6	13.5	0.199765	0.065423
7	15.75	0.16033	0.052733
8	16	0.147903	0.052118
9	15.25	0.111283	0.057955
10	16.25	0.10208	0.05726

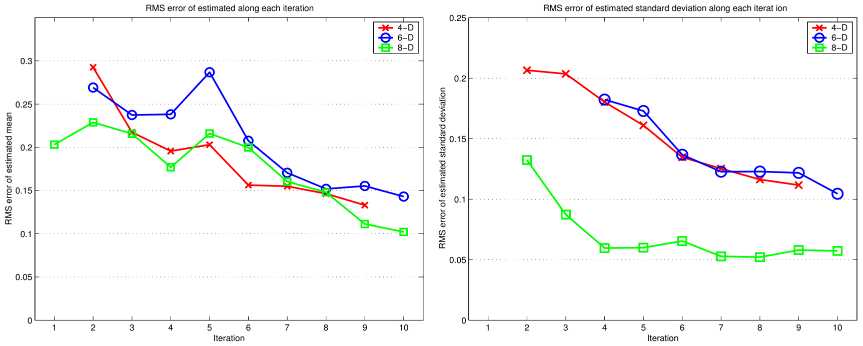


Figure 3.6: RMS of estimated mean and standard deviation along each feedback iteration

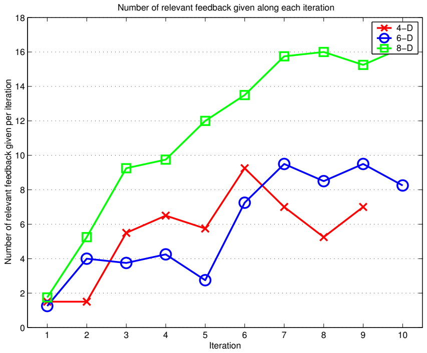


Figure 3.7: Number of feedback given along each feedback iteration

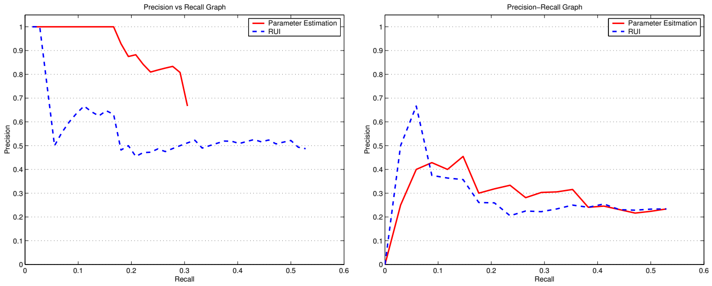


Figure 3.8: Precision-Recall graph of test case in 4-D data - 1

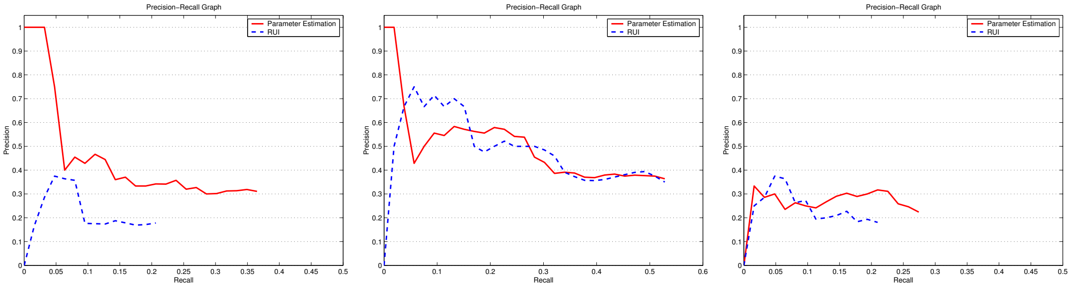


Figure 3.9: Precision-Recall graph of test case in 4-D data - 2

Gaussians with the parameter same as the 4 dimensional case in the convergence experiment. According to the intraweight update equation, we update the weight base on variance of retrieved relevant data points. After several iterations (usually 6 to 7)[4], we use a K-NN (K nearest neighbors) search with weighted Euclidean distance measure to analyze the precision versus recall measure. For our proposed algorithm, as we can estimate the μ and δ of the target distribution, we again perform a K-NN search starting from the mean μ while dropping those data points away from the μ more than 2 δ. Figures 3.8 to 3.11 show the precision versus recall graph for 11 test cases in 4D and 6D data-set. Out of the 11 cases, the parameter estimation method performs better in 5 cases, ties in 4 cases and perform worse in 3 cases when compared to Rui's approach.

[4] This is where the estimated parameters stop converging, please refer to Fig. 3.6.

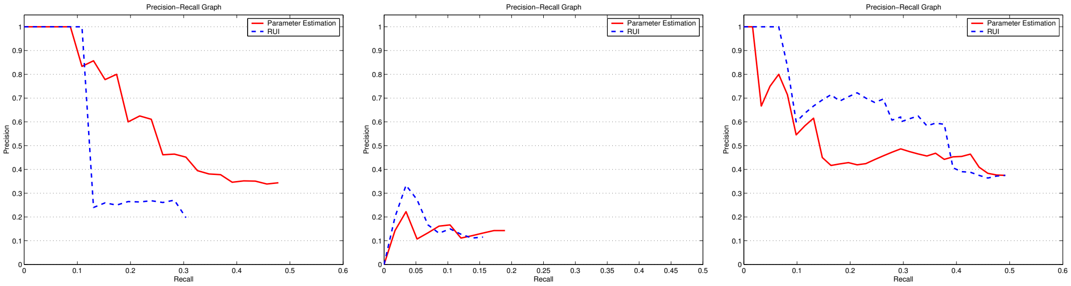


Figure 3.10: Precision-Recall graph of test case in 6-D data - 1

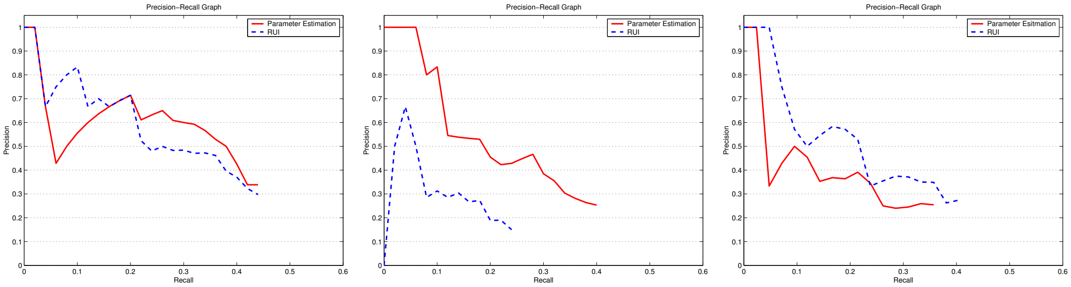


Figure 3.11: Precision-Recall graph of test case in 6-D data - 2

3.3.2 Performance Study in Intra- and Inter- Query Feedback

In this experiment, we work on the Corel image collection, which contains 40,000 images in different categories. Among the 400 categories, we selected 40 categories, each contains 100 images and ranges from the category of **buildings**, **portrait**, **outdoor scenery**, etc. We use the default groupings of images as the ground truth and human knowledge to run automated tests. We use *Color Moment* and *Cooccurence Matrix* as the image features. *Color Moment* computes the mean, variance and skewness values of each color channel in an image. *Cooccurence Matrix* describes the texture of an image by measuring the neighborhood pixel cooccurrence in the gray-level configurations.

The feature vectors of images are first extracted and normalized, then it is used to train a 2D SOM structure of dimension 18×18. Queries are generated and evenly distributed among the 40 classes, while the relevance feedbacks are generated automatically based on the ground truth. All displayed images are marked as relevant if they are in the same class as the query, and the others are marked as non-relevant. The experiment is divided into 3 stages. In the first stage, 160 queries (80 each) are simulated to find out the average recall and precision when using the intra-weight updating version of Rui's approach and the intra-query approach in Section 3.2.4. In the second stage, a number of queries are simulated and the SOM-based intra- and inter-query feedback approach in Section 3.2.3 and 3.2.4 are both used to train the system and retrieve images. In the final stage, 80 queries are simulated again to find out the average recall and precision of our intra-query approach after inter-query feedback is applied. Table 3.5 shows the parameters used in this experiment. We then compare the result of Rui's intra-weight approach and the intra-query approach before and after our SOM-based inter-query feedback training.

Table 3.5: Parameters used in inter-query feedback experiment

Parameter	**Value**
Number of images	4,000
Number of categories	40
Number of iterations used in inter-query feedback	300,500
Number of iterations used in intra-query feedback	9
Ratio of push α_N and pull α_R	20
Feature used	Color Moment (9-dimensions) Cooccurence Matrix (20-dimensions)

Figure 3.12 shows the Recall-Precision graph, averaged among 80 queries, of Rui's approach and our approach before and after the SOM-based training. We make two observations from the experiment. The first one is the intra-query approach performs better after the SOM-based inter-query feedback training. It shows that the retrieval precision can be improved by re-organizing the feature vector space with SOM. The second one is Rui's approach may perform better than our intra-query approach initially, but when more relevant images are retrieved, our performance on precision becomes better. It is because our approach is target for estimating the distribution of the whole category

of relevant images instead of the distribution of relevant images around the query; thus, when retrieving more images, its performance becomes better. Moreover, the experiment indicates that the improvement is sensitive to the push-pull value, their ratio and the number of iterations used in inter-query feedback.

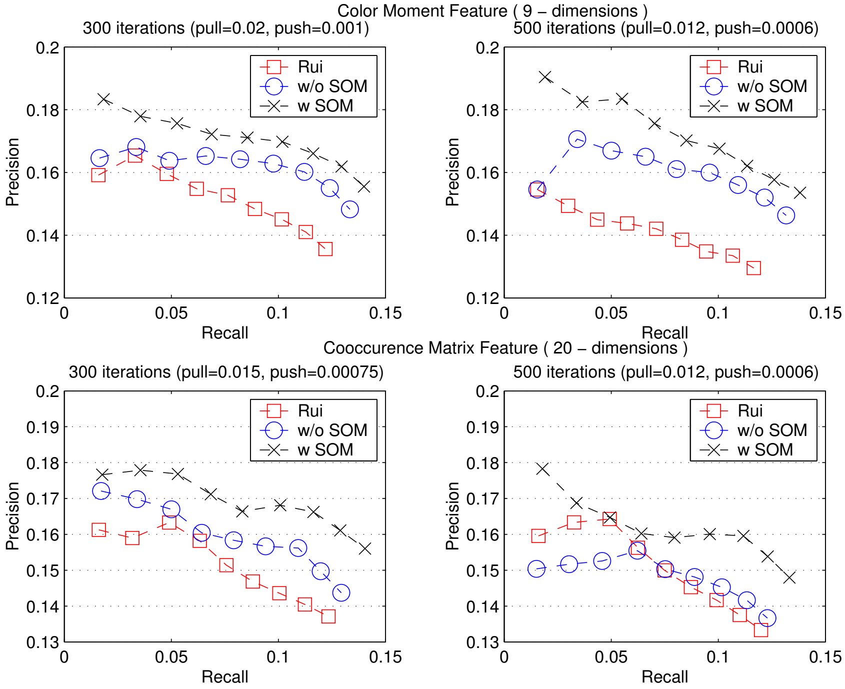


Figure 3.12: Precision-Recall graph showing performance in real data

3.4 Conclusion

In this section, we proposed an approach to estimate a user's target distribution through learning from his feedback via EM algorithm to be applied in CBIR. We have verified the correctness and accuracy of this algorithm through experiment on synthetic data. We proposed a display set selection strategy that utilizes the information given by user in distinguishing image classes in contrast to the K-NN approach for display set selection. Also, our method is based on the parameter estimation of target distribution, we do not need to perform a global update each image in the database accordingly. However, our algorithm also has weaknesses. Since we need to accumulate 3 relevant data points before starting the estimation, users might find this too long[5]. Moreover, maximum entropy strategy is used for display set selection, thus, user might feel that the system cannot present the most relevant data during the feedback process.

As the distribution of image classes in low-level feature space is not well clustered, this makes the estimation of user's target distribution difficult. In the second part, we propose a SOM-based approach for re-organizing the feature space of images using inter-query feedback information. As a result, the distribution of similar images more or less follows a Gaussian-like distribution which make it more efficient for estimation. The proposed approach benefits from reduced data-size and unchanged feature vector dimensionality compared to existing inter-query feedback approaches. We demonstrate the improvement in retrieval accuracy through experiments on real data.

[5] According to data shown in Fig. 3.7, this usually occur in the 2^{nd} iteration.

Chapter 4

DIStributed COntent-based Visual Information Retrieval

When things are distributed, they should be handled in a distributed way, so do content-based image retrieval. When CBIR is to be deployed in a distributed environment, there no longer exists centralized index of feature vectors, how can we locate images with similar content? It is obvious (but wasteful) to query different storage locations for a list of similar images. Instead, we investigate the possibility of forming communities among different storage locations and apply better query scheme to make CBIR more efficient.

4.1 Introduction

As stated in Section 1.1, one of the driving forces of CBIR is the need for databases. We envisage the shared collection among users in a P2P network provides enormous source and storage space for CBIR. We take the challenge to build a CBIR system in a P2P distributed environment. In the process of development, we propose a self-organizing and query routing approach to solve the data-location problem encountered. Section 4.2 and 4.3 explain our approach in improving CBIR in P2P. Section 4.4 details the system architecture of our P2P client program, DISCOVIR. Section 4.5 describes simulation experiments used to verify and analyze our proposal.

4.2 Peer Clustering

The design goal for our strategy is to improve data lookup efficiency in a completely distributed P2P network, while keeping a simple network topology and number of message passings to minimum. In our proposed network, there are two types of connections managed, namely ***random*** and ***attractive*** as shown in Fig. 4.1. Random connections are to link peers that randomly chosen by the users. Attractive connections are to link peers sharing similar data together. We perform clustering at the level of overlaying network topology and also locally shared data, thus content-based query routing is realizable to improve query efficiency; therefore, our algorithm manages to scale when the network grows. We have implemented a beta version of DISCOVIR, built on top of LimeWire [29] with content-based image search function and improved data lookup efficiency.

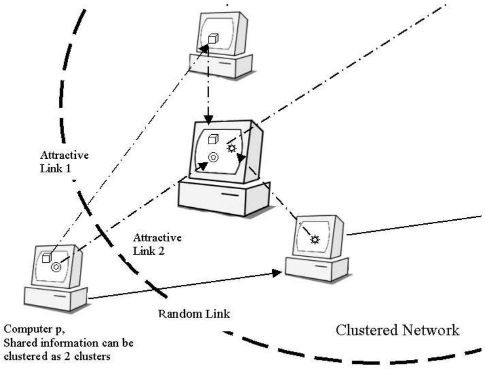


Figure 4.1: Illustration of two types of connections in DISCOVIR

4.2.1 Basic Version

In this section, we describe a restricted (yet effective and insightful) version of our proposed algorithm to illustrate the ideas of peer clustering. Consider a P2P network consisting of peers sharing data of different interest groups as shown in Fig. 4.2. In this network, shared data are classified into three different categories, which are (**C**)omputer, (**H**)istory, and (**S**)ociology respectively. Each peer is represented by one of the three letters above to indicate the majority of documents one shares, and we call this the **signature**.

As shown in the example Fig. 4.2, a new peer of category (**C**)omputer C_4 joins the network by connecting to a randomly selected peer S_3, it sends a signature query to S_3 in the first hop and then propagates to S_1 and H_2 in the second hop; then, to C_1, C_2, H_1, S_2 and S_4 in the third hop and onwards. After collecting the replies from other peers, peer C_4 knows that C_1, C_2 and C_3 share the same type of documents (**C**omputer) with it; thus, C_4 makes an attractive connection to either one of them; in this example, C_3 is chosen. As the network grows, we envision a clustered P2P network is formed. Referring to Fig. 4.2, all peers with the same signature are interconnected by attractive connections, a query routing scheme can thus be applied.

Let us assume the new user of peer C_4 wants to find some documents under the category (**H**)istory. It initiates a query and checks against its signature. It finds that the query mismatches with its signature, thus the query is forwarded through random connection to S_3. S_3 receives the query and performs the same checking. It finds out that they are still mismatched, it then continues to forward the query through random connection to H_2. Once H_2 receives this query, it checks against its signature and finds the query has reached its target cluster. Therefore, it broadcasts the query inside the cluster through attractive connection to H_1. Likewise, when H_1 receives the query, it propagates the query to H_3, and so on. Under this selective query routing scheme, we avoid the query to pass through some unrelated peers, like S_1, C_1. The number of query messages used is reduced while query is still able to reach peers containing the answer.

The detailed description and analysis of this algorithm was proposed in [36], which shows promising result against the conventional Gnutella query mechanism. As this version of peer clustering requires users to assign signature to a peer and agree upon a set of categories in the distributed environment, which is not practical enough to be applied in real world P2P networks, we propose two enhanced versions based on this foundation in the next two sections.

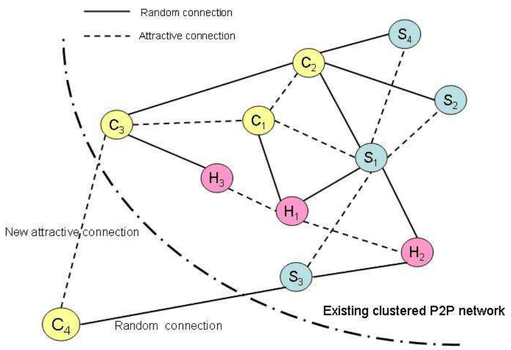


Figure 4.2: Peer clustering - basic version

4.2.2 Single Cluster Version

As peer clustering based on the category of shared documents provided by users explicitly is not practical enough as aforementioned, we move on to peer clustering based on content feature of shared documents that generated automatically. In the information retrieval literature, text documents and images are processed to use feature vectors as their representations, while similarity between two documents are distance measure in the vector space. Clustering in the vector space had been vastly studied to improve retrieval performance by serving as an indexing structure. We surmise such technique is alo valuable to a P2P distributed environment to improve retrieval performance.

In the following, we apply notation in graph theory to model the DISCOVIR network (see Table. 4.1). For the sake of generality, we keep this in high level of abstraction. In this version of peer clustering algorithm, we choose the mean and standard deviation of cluster as signatures of peers, while Euclidean distance is used for similarity measure. In the actual realization, we choose image feature vector as the data structure for representing images, which gives birth to the name of our system DISCOVIR (**DIS**tributed **CO**ntent-based **V**isual **I**nformation **R**etrieval). Here are some definitions:

Table 4.1: Definition of terms used in DISCOVIR

$G\{V, E\}$	The P2P network, with V denoting the set of peers and E denoting the set of connections.
$E = \{E_r, E_a\}$	The set of connections, composed of random connections, E_r and attractive connections, E_a.
$e_a = (v, w,$ $sig_v, sig_w),$ $v, w \in V, e_a \in E_a$	An attractive connection between peers v, w based on sig_v and sig_w.
$\|V\|$	Total number of peers.
$\|E\|$	Total number of connections.
$Horizon(v, t) \subseteq V$	Set of peers reachable from peer v within t hops.
$SIG_v, v \in V$	Set of signatures characterizing the data shared by peer v
$D(sig_v, sig_w),$ $v, w \in V$	Distance measure between specific signatures of two peers v and w.
$Sim(sig_v, q)$ $sig_v \in SIG_v$	Similarity measure between a query q and peer v based on sig_v.
$C = \{C_v : v \in V\}$	The collection of data shared in the DISCOVIR network.
C_v	The collection of data shared by peer v, which is a subset of C.
$REL(c_v, q),$ $c_v \in C_v$	A function determining relevance of data c_v to a query q. 1-relevant, 0-non-relevant.

Definition 1 We consider files shared by a peer can be represented in multi-dimension vectors based on their content, and the similarities among files are based on the distance measure between vectors. Consider

$$f \quad : \quad c_v \rightarrow \vec{c_v} \tag{4.1}$$

$$f \quad : \quad q \rightarrow \vec{q} \tag{4.2}$$

f is the mapping function from file c_v to a vector $\vec{c_v}$. In the notion of image processing, c_v is the raw image data, f is a specific feature extraction method, $\vec{c_v}$ is the extracted feature vector characterizing the image. Likewise, f is also used to map a query q to a query vector $\vec{q}$, to be sent out when user makes a **query by example**[1].

Definition 2 sig_v is the signature representing characteristic of data shared by peer v. We define

$$sig_v = (\vec{\mu}, \vec{\delta}), \tag{4.3}$$

where $\vec{\mu}$ and $\vec{\delta}$ are the statistical mean and standard deviation of the collection of data, shared by peer v, in their feature vector space. From now on, sig_v characterizes the data shared by peer v.

Definition 3 $D(sig_v, sig_w)$ is defined as the distance measure between sig_v and sig_w, in other sense, the similarity between two different peers v and w. It is defined as,

$$D(sig_v, sig_w) = ||\vec{\mu_v} - \vec{\mu_w}||. \tag{4.4}$$

$||\vec{\mu_v} - \vec{\mu_w}||$ is the Euclidean distance between two cluster centroids symbolized by sig_v, sig_w. We define the data similarity of two peers by this formula and use it to help organizing the network.

Based on the above definitions, we introduce a peer clustering algorithm, to be used in the network setup stage, that helps building the DISCOVIR as a

[1] In content-based Image Retrieval, users usually provide an example images or a sketch to search for similar images.

self-organized network oriented in content similarity. It consists of three steps as follows:

1. **Signature Calculation**–Every peer preprocesses its data collection and calculates signature sig_v to characterize its data properties by assuming all feature vectors as a single cluster. Whenever the shared data collection, C_v, of a peer changes, the signature is updated accordingly.

2. **Neighborhood Discovery**–After a peer joins the DISCOVIR network by connecting to a randomly selected peer in the network, it broadcasts a signature query message, similar to a ping-pong message in Gnutella, to ask for signatures of peers within its neighborhood, $sig_w, w \in Horizon(v, t)$. This task is not only done when a peer first joins the network, it repeats every certain interval in order to maintain the latest information on other peers. A more efficient way of signature query is described in Appendix B.

3. **Attractive Connection Establishment**–After acquiring the signature of other peers, one can reveal the peer with signature closest to its according to Eq. 4.4, and initiates an attractive connection to link them up. This attractive connection is reestablished to the second closest one in the host cache[2] whenever current connection breaks.

Fig 4.3 illustrates a working example of single cluster version peer clustering model. The number on each node indicates its time of appearance in the DISCOVIR network, 1 being the oldest node and 11 being the newest node. the (x, y) value besides a node is the mean of its signature in 2-D feature vector space. Table 4.2 is a matrix showing the pair-wise distance of signatures among the peers. When peer 11 joins the network, it first connects to peer 9 by a random connection. It sends out a signature query message to peer

[2] The host cache is being obtained in the neighborhood discovery stage.

Table 4.2: Distance between signatures of peers

peer	1	2	3	4	5	6	7	8	9	10	11
1	0.00	**8.18**	7.24	**3.68**	**0.94**	1.24	4.19	7.61	**0.50**	3.54	3.05
2		0.00	**1.22**	4.83	8.71	9.15	4.63	**0.61**	8.64	4.95	5.48
3			0.00	3.73	7.69	8.14	3.47	0.67	7.70	3.86	4.40
4				0.00	4.0	4.44	**0.61**	4.22	4.17	**0.14**	0.67
5					0.00	**0.45**	4.42	8.12	1.03	3.86	3.33
6						0.00	4.86	8.56	1.14	4.3	3.77
7							0.00	4.02	4.69	0.73	1.14
8								0.00	8.06	4.34	4.88
9									0.00	4.1	3.55
10										0.00	**0.54**
11											0.00

9; after that, it turns out that peer 10 is closest to peer 11 in distance between their signatures. Peer 11 then makes an attractive connection to peer 10. Having all peers joining the DISCOVIR network perform the three tasks described above, you can envision a P2P network with self-organizing ability to be constructed. Peers sharing similar/same content are grouped together like a social community. The detail steps of peer clustering is illustrated in Algorithm 1.

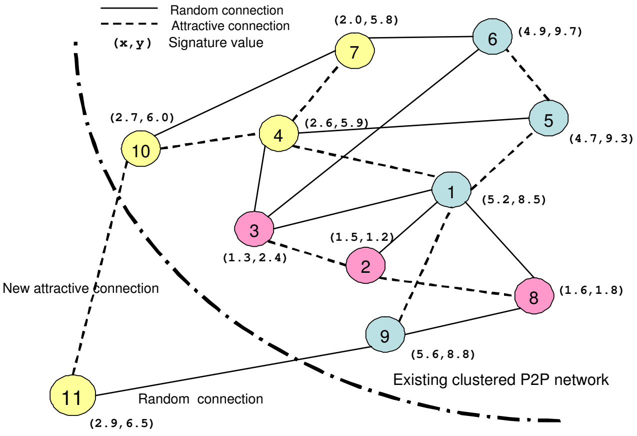


Figure 4.3: Peer clustering - single cluster version

4.2.3 Multiple Clusters Version

The previous version of peer clustering assumes majority of data shared by a peer falls in the same category, e. g. , a peer shares collection of sunset

Algorithm 1 Algorithm for Peer Clustering - Single Cluster Version

```
SC-Peer-Clustering(peer $v$, integer $ttl$)
for all $w \in Horizon(v, ttl)$ do
  Compute $D(sig_v, sig_w)$
end for
$E_a = E_a \cup (v, w, sig_v, sig_w)$ having $min(D(sig_v, sig_w))$
```

images or collection of computer science research papers; thus the extracted feature vectors might be clustered together and one signature is able to describe the data characteristic reasonably. However, most users share documents of various topics in real-world situation as illustrated in Fig. 4.4. There should be an attractive connection between sub-cluster A_2 of peer A and sub-cluster B_3 of peer B although their cluster centroid are far apart.

We propose to a set of signatures, SIG_v, to represent a peer because the document collection is likely to fall into several categories and their extracted feature vectors form several clusters as well. With these changes, peers may establish several attractive connections depending on the number of local sub-clusters. The revised algorithm is illustrated in Algorithm 2 and two changes in the main steps are updated as follows:

1. **Signatures Calculation**–In the preprocessing stage, a set of signatures SIG_v is calculated, which are the statistical mean and standard deviation of sub-clusters found using some clustering algorithm like K-means [2], competitive learning [44], or expectation maximization [15]. The number of signatures is variable and it is **a trade-off between resolution of cluster and computational cost**. In the implementation of DISCOVIR, we choose to use competitive learning for the sake of low computational cost.

2. **Attractive Connection Establishment**–This process is the same as that in previous section except we make attractive connection for every signature a peer possess. In case the number of signatures exceeds the

number of attractive connections available to be established, the standard deviation in a signature is used to determine the quality of that attractive connection. The smaller the standard deviation, the better to make attractive connection because it implies a dense cluster.

We have introduced the final and most practical version of peer clustering algorithm in DISCOVIR, having all peers follow this procedure when building the network, a self-organizing network is formed. We delineate a query routing strategy in the next section to improve retrieval performance.

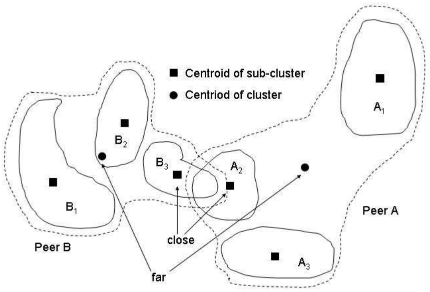


Figure 4.4: Multiple clusters exist in a peer

Algorithm 2 Algorithm for Peer Clustering - Multiple Clusters Version

```
MC-Peer-Clustering(peer $v$, integer $ttl$)
for all $sig_v \in SIG_v$ do
   for all $w \in Horizon(v, ttl)$ do
      for all $sig_w \in SIG_w$ do
         Compute $D(sig_v, sig_w)$
      end for
   end for
   $E_a = E_a \cup (v, w, sig_v, sig_w)$ having $min(D(sig_v, sig_w))$
end for
```

4.3 Firework Query Model

To make use of our clustered P2P network, we propose a content-based query routing strategy called Firework Query Model (FQM). In this model, a query message is routed selectively according to its content. Once it reaches its designated cluster, the query message is broadcasted by peers through attractive connections inside the cluster much like an exploding firework as shown in Fig. 4.5. Our strategy aims to:

1. Minimize the number of message passings in the overlay network,
2. Reduce the workload of each computer, and
3. Maximize the number of relevant documents retrieved from the P2P network.

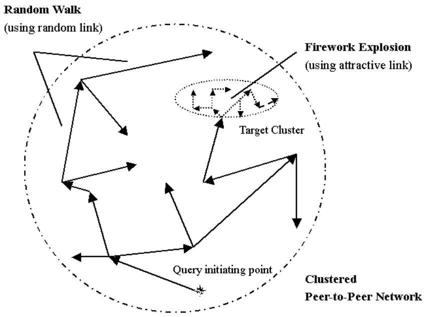


Figure 4.5: Illustration of firework query

Here, we introduce the algorithm to determine how a query message is routed and propagated like a firework in Algorithm 3. When a peer receives the query, it carries out two steps:

1. **Shared File Look Up**–The peer looks up its shared documents for those matched with the query. Let q be the query, and $\vec{q}$ be its vector representation, $REL(c_v, q)$ is the relevance measure between the query and the document c_v shared by peer v. It depends on a L_2 norm defined as,

$$REL(c_v, q) = \begin{cases} 1 & ||\vec{c_v} - \vec{q}|| \leq T \\ 0 & ||\vec{c_v} - \vec{q}|| > T, \end{cases}$$

where T is a threshold defining the degree of result similarity a user wants. If it is larger, then more results will be obtained (but no guarantee to maintain high precision). For any shared documents within the matching criteria of the query, the peer replies the requester. In addition, we can reduce the number of $REL(c_v, q)$ computations by performing local clustering[3] in a peer, thus speeding up the process of query response.

2. **Route Selection**–The peer calculates the similarity between the query and each signature of its local clusters, sig_v , by the following equation:

$$Sim(sig_v, q) = \prod_{i}^{d} e^{-\frac{(q_i - \mu_i)^2}{2\delta_i^2}}, \quad sig_v = (\vec{\mu}, \vec{\delta}), \tag{4.5}$$

where i is the index for dimension of feature vector. If none of the distance measures between its local clusters' signatures and the query, $Sim(sig_v, q)$, is larger than a preset threshold, θ, the peer will propagate the query to its neighbors through random connections. Otherwise, if one or more $Sim(sig_v, q)$ is larger than the threshold, it implies the query has reached its target cluster. Therefore, the query will be propagated through corresponding attractive connections much like an exploding firework.

In our model, we retain two existing mechanisms in Gnutella network for preventing query messages from looping forever in a distributed network, namely, the Gnutella replicated message checking rule and Time-To-Live (TTL) of messages. When a new query appears to a peer, it is checked against a local cache for duplication. If it is found that the same message has passed through before, the message will not be propagated[4]. The second mechanism is the use of TTL value to indicate how long a message can survive. Similar to IP packets, every Gnutella messages are associated with a TTL. Each time when

[3] We compare query to sub-clusters first instead of comparing with each documents.

[4] This is done by adding a Globally Unique ID, GUID for every message sent out.

s message passes through a peer, the TTL value is decremented by one. Once the TTL reaches zero, the message is dropped and no longer forwarded. There is a modification on DISCOVIR query messages from the original Gnutella messages. In our model, the TTL value is decremented by one with a different probability when the message is forwarded through different types of connection[5]. For random connections, the probability of decreasing TTL value is 1. For attractive connections, the probability of decreasing TTL value is an arbitrary value in [0, 1] called Chance-To-Survive (CTS). This strategy reduces the number of message passings outside the target cluster, while more relevant information can be retrieved inside the target cluster because the query message has a greater chance to survive depending on how large the CTS value is. In particular, we choose CTS=1 in our simulation experiment.

Algorithm 3 Algorithm for the Firework Query Model

```
Firework-query-routing (peer $v$, query $q$)
for all $sig_v \in SIG_v$ do
  if $Sim(sig_v, q) > \theta$ (threshold) then
    if $rand() > CTS$ then
      $q_{ttl} = q_{ttl} - 1$
    end if
    if $q_{ttl} > 0$ then
      propagate $q$ to all $e_a(a, b, c, d)$ where $a = v, c = sig_v$ or $b = v, d = sig_v$
      (attractive link)
    end if
  end if
end for
if Not forwarding to attractive link then
  $q_{ttl} = q_{ttl} - 1$
  if $q_{ttl} > 0$ then
    forward $q$ to all $e_r(a, b)$ where $a = v$ or $b = v$ (random link)
  end if
end if
```

[5] Similar method of propagating queries with different probability was studied in [4] from the percolation theory perspective.

4.4 Implementation and System Architecture

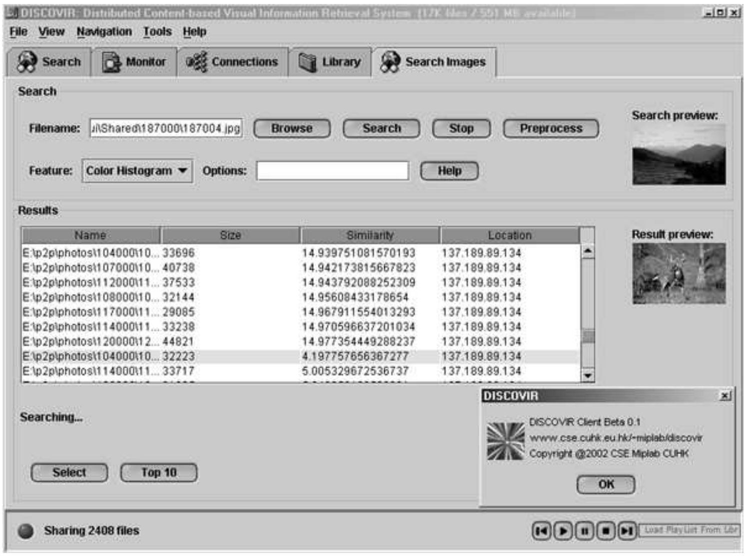
We briefly outline the process of sharing and retrieving image in DISCOVIR. First, each peer is responsible for extracting features, e.g., color, texture, shape, etc, from its shared images using the DISCOVIR client program. Using this program, each peer maintains the local index of feature vectors of its image collection. When a peer, the requester, initiates a query by giving an example image using a particular feature extraction method, it first performs feature extraction on the example image and sends the feature vector, contained in a query message, to all its connecting peers. Consequently, other peers compare this query to their feature vector index. Based on a distance measure, they find a set of similar images, if exists, and return results back to the requester. Likewise, these peers will propagate the query to their connecting peers using the Firework Query Model described in previous section and this process continues to query other peers in the network.

DISCOVIR uses a plug-in architecture to support different feature extraction method. User may select to download a plug-in in the form of compiled Java bytecode if they want to perform CBIR based on that particular feature extraction method. The following is a screen capture of the DISCOVIR client program, see Fig. 4.6, which can be downloaded from this site [32]. In the following, we describe the overall architecture of DISCOVIR, which operates on the current Gnutella network, and the modification of query messages.

4.4.1 Gnutella Message Modification

The DISCOVIR system is compatible to the Gnutella (v0.4) protocol [18]. In order to support the image query functionalities mentioned, two types of messages are added. They are:

- **ImageQuery** - A special type of the Query message. It is to carry name of feature extraction method and feature vector of query image,



Filename: ...\Shared\187000\187004.jpg

Feature: Color Histogram

Options:

Name	Size	Similarity	Location
E:\p2p\photos\104000\10...	33696	14.939751081570193	137.189.89.134
E:\p2p\photos\107000\10...	40738	14.942173815667823	137.189.89.134
E:\p2p\photos\112000\11...	37533	14.943792088252309	137.189.89.134
E:\p2p\photos\108000\10...	32144	14.95608433178654	137.189.89.134
E:\p2p\photos\117000\11...	29085	14.967911554013293	137.189.89.134
E:\p2p\photos\114000\11...	33238	14.970596637201034	137.189.89.134
E:\p2p\photos\120000\12...	44821	14.977354449288237	137.189.89.134
E:\p2p\photos\104000\10...	32223	4.197757656367277	137.189.89.134
E:\p2p\photos\114000\11...	33717	5.005329672536737	137.189.89.134

Searching...

DISCOVIR

DISCOVIR Client Beta 0.1
www.cse.cuhk.eu.hk/~miplab/discovir
Copyright @2002 CSE Miplab CUHK

Sharing 2408 files

Figure 4.6: Screen-shot of DISCOVIR

see Fig. 4.7.

- **ImageQueryHit** - A special type of the QueryHit message. It is to respond to the ImageQuery message, it contains the location, filename, size of similar images retrieved, and their similarity measure to the query. Besides, the storage location information of corresponding thumbnails are added for the purpose of previewing result set at a faster speed, see Fig. 4.8.

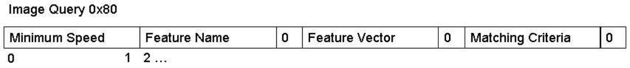


Image Query 0x80

Minimum Speed	Feature Name	0	Feature Vector	0	Matching Criteria	0
0 1	2 ...					

Figure 4.7: ImageQuery message format

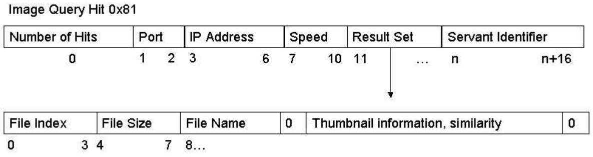


Image Query Hit 0x81

Number of Hits	Port	IP Address	Speed	Result Set	Servant Identifier
0	1 2	3 6	7 10	11 ...	n n+16

File Index	File Size	File Name	0	Thumbnail information, similarity	0
0 3	4 7	8...			

Figure 4.8: ImageQueryHit message format

4.4.2 Architecture of DISCOVIR

In this section, we describe the architecture of a DISCOVIR client and the interactions between modules in order to perform CBIR in a P2P network. Figure 4.9 depicts the key components and their interactions a DISCOVIR client. As DISCOVIR is built based on the LimeWire [29] open source project, the operations of Connection Manager, Packet Router and HTTP Agent remain nearly unchanged but with some additional functionalities to improve the query mechanism originally used in Gnutella network. The Plug-in Manager, Feature Extractor and Image Indexer are newly added to support the CBIR function. The User Interface is modified to incorporate the image search panel. Figure 4.6 shows a screen capture of DISCOVIR in the image search page. Here are brief descriptions of the six major components:

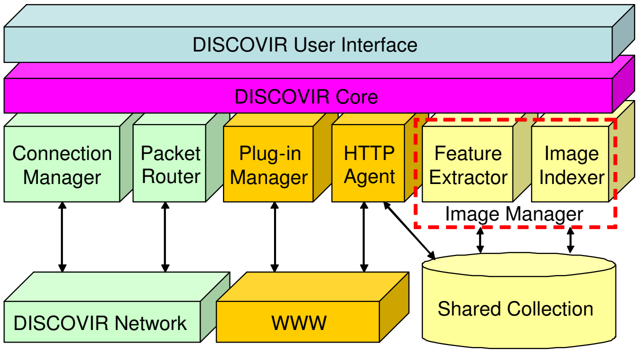


Figure 4.9: Architecture of DISCOVIR

- **Connection Manager** - It is responsible for setting up and managing TCP connections between DISCOVIR clients.
- **Packet Router** - It controls the routing, assembling and disassembling of message between DISCOVIR network and different components in a peer.
- **Plug-in Manager** - It coordinates the download and storage of different feature extraction plug-ins and their interactions with Feature Extractor

and Image Indexer.

- **HTTP Agent** - It is a tiny web-server that handles file download requests from other DISCOVIR peers using HTTP protocol.
- **Feature Extractor** - It collaborates with the Plug-in Manager to perform various feature extractions and thumbnail generations of the shared image collection. It is involved in two main functions:
  - **Preprocessing** - It extracts the feature vectors of shared images in order to make its collection being searchable in the network.
  - **Real Time Extraction** - It extracts the feature vector of query image on-the-fly and passes the query to Packet Router.
- **Image Indexer** - It indexes the image collection by content feature and carries out clustering to speed up the retrieval of images.

4.4.3 Flow of Operations

The four main steps of performing CBIR in DISCOVIR are listed in detail as follows.

1) Preprocessing

The Plug-in Manager module is responsible for contacting the DISCOVIR website to inquire a list of available feature extraction modules. It downloads and installs selected modules upon user's request. Currently, DISCOVIR supports various feature extraction methods in color and texture categories such as AverageRGB, GlobalColorHistogram, ColorMoment, Co-occurrence matrix, etc. All feature extraction modules strictly follow a predefined API in order to realize the polymorphism of switching between different plug-ins dynamically.

The Feature Extractor module extracts feature and generates thumbnails for all shared images using a particular feature extraction method chosen by a user. The Image Indexer module then indexes the image collection using the extracted multi-dimensional feature vectors. Compared with the centralized web-based CBIR approach, sharing the workload of this computational costly task among peers helps solving the bottle-neck problem by utilizing distributed computing resources.

2) Connection Establishment

For a peer to join the DISCOVIR network, the Connection Manager module asks a bootstrap server, which is a program maintaining a list of peers available for accepting income connection currently in the network. Once an IP address is known, the peer hooks up to the DISCOVIR network by connecting to currently available peers.

3) Query Message Routing

When a peer initiates a query for similar images, the Feature Extractor module processes the query image instantly and assembles an ImageQuery message, as shown in Fig. 4.7, to be sent out through Packet Router module. Likewise, when other peers receive the ImageQuery messages, they need to perform two operations, **Query Message Propagation** and **Local Index Look Up**.

- **Query Message Propagation** - When propagating the query message, the Packet Router module of DISCOVIR employs two checking rules adopted from Gnutella network in order to prevent messages from looping forever in the DISCOVIR network. They are (1) Gnutella replicated message checking rule and (2) TTL of messages respectively. The replicated message checking rule prevents a peer from propagating the same query message again. The TTL mechanism constrains the reachable horizon of a query message. In addition, Firework Query Model is applied to determine how to route a query.

- **Local Index Look Up** - The peer searches its local index of shared files for similar images using the Image Indexer module and information in the *ImageQuery* message. Once similar images are retrieved, the peer delivers an *ImageQueryHit* message, see Fig. 4.8, back to the requester through Packet Router module.

4) Query Result Display

When an *ImageQueryHit* message returns to the requester, user obtains a list detailing the location and size of matched images. In order to retrieve the query result, the HTTP Agent module downloads thumbnails, generated in the preprocessing stage, or full size image from another peer using HTTP protocol. On the other hand, HTTP Agent modules in other peers serve as web servers to deliver the requested images.

4.5 Experiments

In this section, we discuss the design of simulation experiments and evaluate the performance of our proposed Peer Clustering and Firework Query Model. First, we present our model of P2P network used in the simulation and introduce the performance metrics used for evaluation in section 4.5.1. We study the performance of FQM using different parameters, the number of peers in Section 4.5.2, the TTL of query message in Section 4.5.3, and different data distribution in Section 4.5.4. We show how our strategy performs and behaves at scale and give final remarks at the end.

4.5.1 Simulation Model of the Peer-to-Peer Network

Our goal of the experiment is to model a typical Peer-to-Peer network of nodes T where each node contains a set of documents. We built different size of

P2P networks to evaluate the performance of different search mechanisms. As shown in Figure 4.10, the network follows a power law distribution with an average degree of 3.97 when the number of peers is 8000. The right one is a snapshot of Gnutella graph captured on Aug 2001 obtained from [30], which follows a two stages power law distribution. The number of peers in each network varies from 2000 to 20000. The diameter[6] varies from 9 to 11, and the average distance[7] between two peers varies from 5.36 to 6.58. Table. 4.3 lists the detailed information of each model.

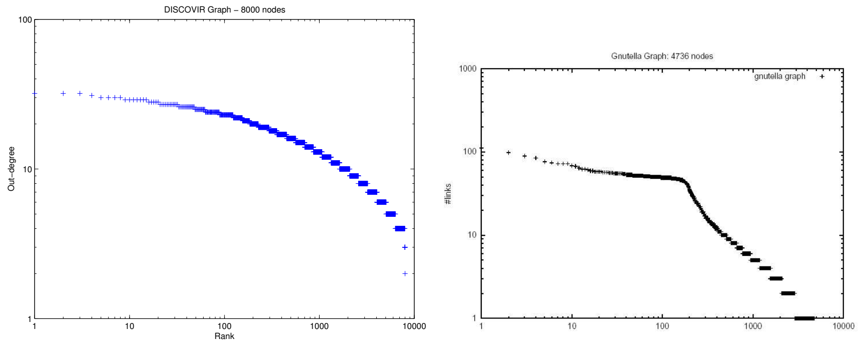


Figure 4.10: Network characteristic of Gnutella (right) and our simulation model (left)

Table 4.3: Characteristics of each P2P network model

Number of Peer	2000	4000	8000	12000	16000	20000
Diameter	9	9	10	10	11	11
Average distance between two peers	5.36	5.77	6.12	6.33	6.47	6.58

The simulations are done over different network configurations and data placement methods. We run 50 iterations and average the results for each set of parameters. For every 10 iterations, we rebuild the simulated network again, initiate a query starting from a randomly selected peer and collect the statistical information listed in next section. We use the images from the Corel image collection CD as the real data in our experiment. We randomly assign

[6] It is the largest number of hops among the shortest path between any pairs of peers.

[7] The average shortest path between any pairs of peers.

different classes of images to each peer; therefore, the data placement in every 10 iterations is totally different. Color Moment is chosen as the feature vector to represent images. Competitive Learning [44] clustering algorithm is used to cluster data insides each peer.

We use synthetic and real datasets of different data distribution characteristic in our experiment in order to analyze the effect of data distribution on the performance. Table 4.4 shows statistics about the data distribution. Both real and synthetic dataset contain 100 classes. We treat each class as one single cluster and calculate the pairwise Euclidean distances between cluster centroids. The dimension of dataset is 9, The variances in each dimension are averaged for each cluster, and this is used as a measure of cluster denseness. Fig. 4.11 shows a SOM visualization of the two datasets. We realize that clusters of synthetic dataset are more separated apart and denser, while real dataset clusters are overlapped and sparse. This might be a justification for the better performance in synthetic dataset as shown in later sections, because we use distance between centroids of clusters as the criteria for peer clustering shown in Eq. 4.4 and the query routing scheme, Eq. 4.5,reduces network traffic more if clusters are denser.

Table 4.4: Data distribution of real and synthetic data

Inter-cluster distance			Mean of variances		
	real data	synthetic data		real data	synthetic data
max	1.4467	1.8207	max	0.0153	0.0128
min	0.0272	0.3556	min	0.0006	0.0042
avg	0.3298	1.1159	avg	0.0112	0.0086

Performance Metrics

The following metrics are used to evaluate the performance:

1. **Recall** [3]– The power of a search strategy to return desired results. It is the fraction of relevant documents in the P2P network being retrieved

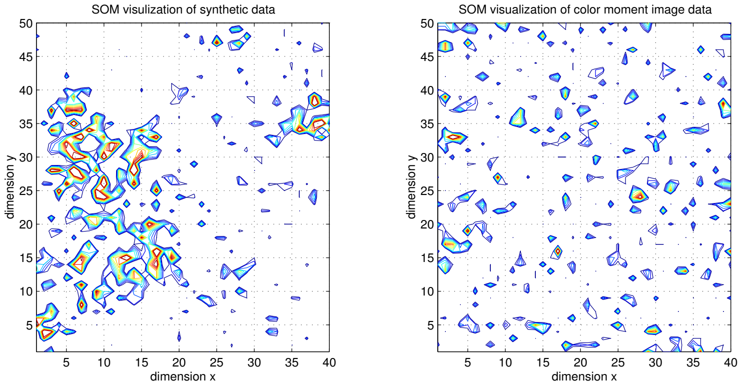


Figure 4.11: SOM visualization of distribution of synthetic and real data

i.e., recall $= \frac{|R_a|}{|R|}$ where R_a is the set of retrieved relevant documents, R is the set of relevant documents in the P2P network. If recall is high, it means more relevant documents can be retrieved i.e., the performance is better, provided that the query is not flooding the network.

2. **Query scope**– The fraction of peers being visited by each query i.e., query scope $= \frac{|V_a|}{|V|}$ where V_a is the set of peers having received and handled the query, V is the set of all peers in the P2P network. For each query, if the fraction of peers involved is lower, the system is more scalable, provided that a significant amount of desired result is retrieved.

3. **Query efficiency**– The ratio between the recall and query scope i.e., query efficiency $= \frac{\text{recall}}{\text{query scope}}$. An query scheme is more desirable if we can retrieve more relevant documents but only visit few peers. If the inherent Gnutella search mechanism is used, the query efficiency is equal to 1. In other words, the more peers a query visited, the more relevant results retrieved.

4. **Query message utilization**– As the number of query messages is proportional to the query scope (number of peers visited), it is useless to investigate it again. Instead, we figure out that the ratio of number of query

messages generated to the number of peers visited is more useful in evaluating the performance i.e., utilization=$\frac{\text{Number of query meesages used}}{\text{Number of peers visited}}$. The lower the ratio, the better the utilization as fewer messages are generated to visit the same number of peers.

5. **Reply path length**– Reply path length is the number of hops between a query requester and a peer that replies the query. In our experiments, it is defined as the average of reply path length of the first ten repliers. It is more desirable if the minimum reply path length is shorter; thus, the requester can receive the reply in a shorter time.

4.5.2 Number of Peers

This experiment tests the scalability of our search mechanism. The number of peers in each network varies from 2000 to 20000. The experiment parameters are listed in Table. 4.5.

Table 4.5: Experiment parameters for scalability experiment

Number of Peer	2000 - 20000
Topology if the P2P network	Power law distribution with an average branching degree of 1.97
Diameter of the P2P network	9 - 11 hops
Average distance between 2 peers	5.4 - 6.6 hops
Number of documents assigned to each peer	100 documents (1 class)
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	7

Recall

Figure 4.12 shows the recall against number of peers in two search mechanisms and data-sets. When the size of network increases, Firework Query Model (FQM) maintains a relatively high recall value, while the recall for Breadth First Search (BFS) drops when size of network grows. We conclude that the power to retrieval relevant data in the network of our algorithm is insensitive to the change of network size; thus, our mechanism is more scalable. Even the size of network grows, FQM still can reach a large portion of the network containing the query target. We observe that recall of FQM for synthetic data is comparatively lower. This observation implies that peers containing the same category of synthetic data might be separated into several isolated sub-communities. As a result, a query can only reach and be broadcasted in one or few of them. According to the data distribution of synthetic data described in Section 4.5.1 and the equation, Eq. 4.4 used in clustering peers, the chance for peers belonging to different categories are unlikely to form attractive connection in the synthetic dataset; thus, the chance for forming isolated sub-communities is higher.

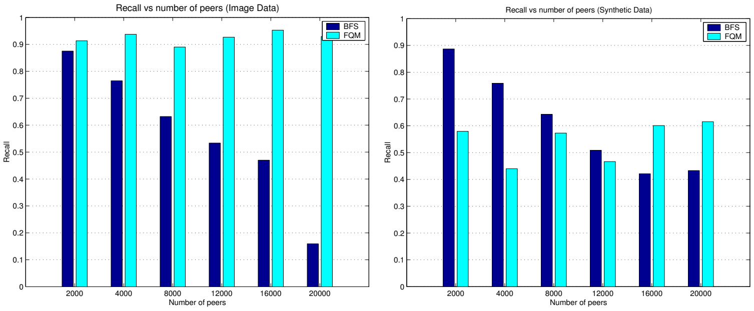


Figure 4.12: Recall against number of peers (left: real, right: synthetic)

Query scope

As seen in Figure 4.13, when the network grows, the percentage of peer visited by query message in BFS decreases. Since the TTL is fixed, the number of peer visited cannot increase unlimitedly, so this percentage drops when number of peers in the network increases. For FQM, the visited peer percentage for both real and synthetic data remains more or less constant, this is because TTL of query message is not decremented through attractive connections. The visited peer percentage of image data is much higher than that of synthetic data because the route selection formula, Eq. 4.5, favors queries to pass through attractive connections if signatures of peers are close and overlapped, and the clusters' densities are sparse.

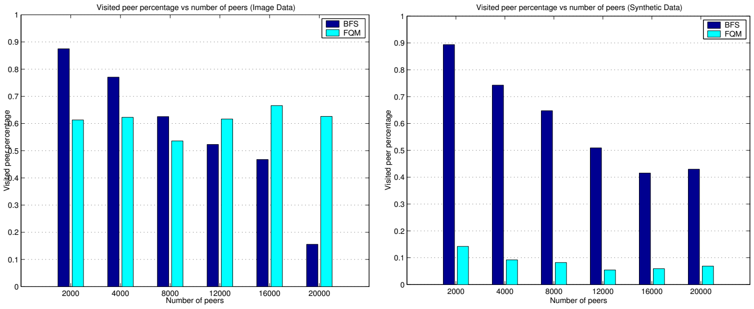


Figure 4.13: Query scope against number of peers (left: real, right: synthetic)

Query efficiency

Neither recall nor query scope can account for the performance of a query scheme alone, as one may get high recall but visit all peers in the network, while one may visit few peers but only retrieve a few relevant documents. Thus, query efficiency is introduced to reveal the actual performance of a query scheme. As shown in Figure 4.14, efficiency of FQM outperforms BFS more than 3 times at most. The curve of FQM follows a small bell shape. Query efficiency exhibits such behavior due to two reasons:

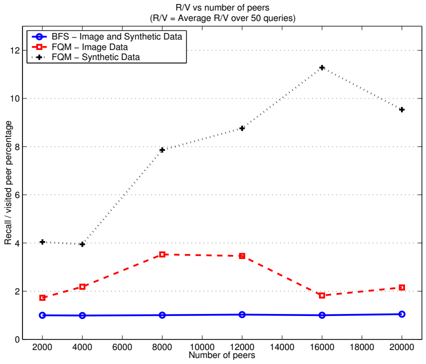


Figure 4.14: Query efficiency against number of peers

1. The network can be clustered more appropriately when the network size increases. In other words, the effect of forming communities getting more noticeable when the network grows.

2. When the network size increases further, a query might not reach its target cluster for a low TTL value (The TTL used is 7 in this experiment and the diameter of network is 11 when number of peers is 16000 and 20000), so query efficiency starts to drop.

Therefore, choosing a good TTL value is important in our algorithm and this will be discussed in the next section. Note that the average query efficiency shown in this graph is **NOT** $\frac{\text{average recall}}{\text{averge query scope}}$, but the averaged query efficiency of each iteration.

Query message utilization

The ratio in Figure 4.15 can be interpreted as the number of query messages used in order to visit one peer. FQM performs better than BFS in both real and synthetic data.

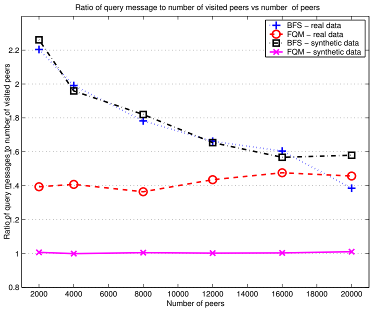


Figure 4.15: Query message utilization against number of peers

Average reply path length

Figure 4.16 shows the average reply path length. On average, the relevant documents are 3-7 hops away in BFS, however, the path length in FQM is just

2-5 hops. It is more desirable if the average reply path length is shorter. The requester can receive replies in a shorter time.

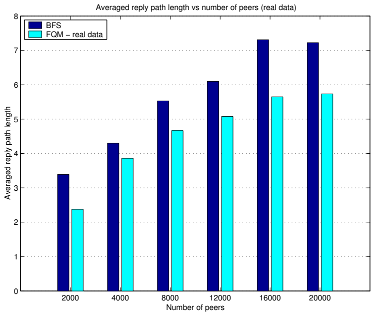


Figure 4.16: Averaged reply path length against number of peers

4.5.3 TTL of Query Message

In this section, we explore how the performances are affected by different Time-To-Live (TTL) values of query message. The number of peers in each network is fixed to 10000. The TTL value varies from 4 to 9. The experiment parameters are listed in Table. 4.6.

Table 4.6: Experiment parameters for TTL experiments

Number of Peer	10000
Topology if the P2P network	Power law distribution with an average branching degree of 1.97
Diameter of the P2P network	10 hops
Average distance between 2 peers	6.2 hops
Number of documents assigned to each peer	100 documents (1 class)
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	4 - 9

Recall

Fig. 4.17 shows the recall when TTL of query messages changes. As expected, when TTL increases, more peers are visited, thus recall increases. Both graphs show this expected trend, while FQM attains a higher recall at low TTL because TTL is not decremented through attractive connection, making it to reache more peers. Again, FQM of synthetic data shows a comparatively lower recall, which is due to the isolated communities effect.

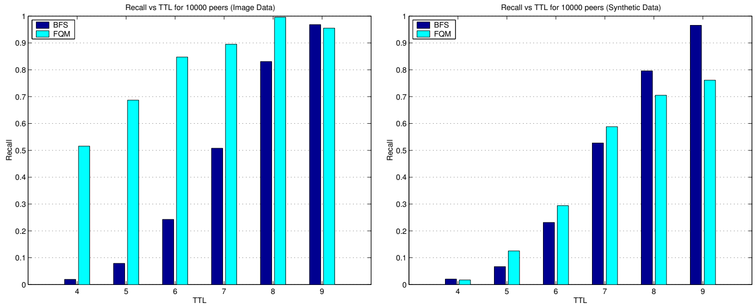


Figure 4.17: Recall against TTL value of query message (left: real, right: synthetic)

Query scope

Fig. 4.18 shows the visited peer percentage when TTL of query messages increases. Again, when TTL increases, more peers are visited. For real data, visited peer percentage continue to increase for BFS, while FQM also increases but at a lower rate. This is the same for synthetic data but FQM visits much less number of peers because synthetic data is well clustered. From this observation, it shows that FQM solves the problem of query message broadcasting and successfully reduces number of peers visited, while maintaining an acceptable recall.

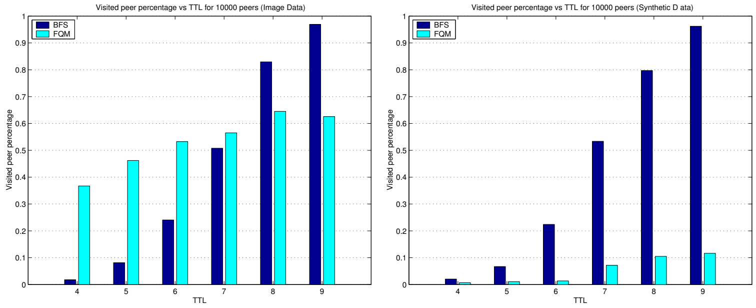


Figure 4.18: Query scope against TTL value of query message (left: real, right: synthetic)

Query efficiency

As shown in Figure 4.19, FQM outperforms BFS under different TTL values of query message. We found that the optimal TTL value is 7 in a network size of ten thousands peers under FQM. The query efficiency is low at the beginning because the TTL value is not large enough for a query message to reach its target cluster. When the TTL increases, the query has a higher chance to reach its target cluster, therefore, the query efficiency increases. When the TTL is 7, the query efficiency is optimal because the query message can just reach its target cluster without further visiting non-relevant peers. However, further increasing the TTL value only generates unnecessary traffic; therefore, the query efficiency starts to drop when TTL is 8.

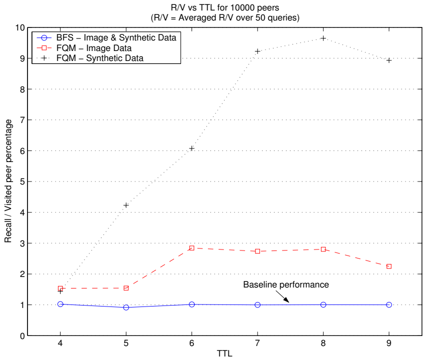


Figure 4.19: Query efficiency against TTL value of query message

Query message utilization

In Figure 4.20, it shows that query message utilization drops (the ratio increases) when TTL of query message increases. This is because larger TTL makes a query message visit the same peer through two different path; thus, the ratio get higher. The ratio of BFS increases with TTL at a higher rate than that of BFS, this implies that when TTL is being increased to improve recall, BFS becomes less efficient than FQM.

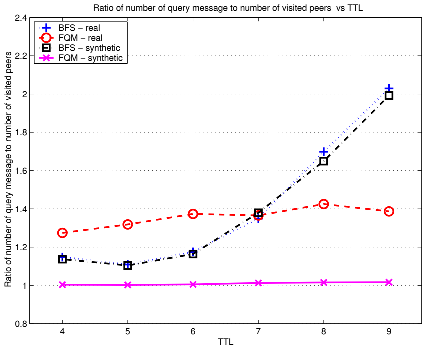


Figure 4.20: Query message utilization against TTL value of query message

4.5.4 Effects of Data Resolution on Query Efficiency

As indicated in previous sections, the query efficiency for synthetic data always performs better than that of real data. We assume that this improvement results from that fact that clusters in synthetic data are denser and well separated apart. In this experiment, we investigate the effect of cluster size and their inter-cluster distances on query efficiency using real data. We assign 2-4 classes of image (each containing 100 documents) to each peer. Using competitive learning method, each peer is represented by 1 or 3 signatures in two sets of experiment respectively. Obviously, peers represented by 1 signature are unable to describe their shared images accurately enough; thus, the clusters formed are sparse and overlapped. While the clusters formed from peers represented by 3 signatures are denser and more separated. With this configuration, we investigate how query efficiency can be improved. Table 4.7 and Fig. 4.21 show the details and result of this experiment.

According to Fig. 4.21, FQM outperforms BFS in both configurations. For FQM using 3 signatures per peer, it outperforms that of 1 signature per peer in all configurations of number of peers or TTL. Based on this observation, we conclude that our peer clustering and firework query model is sensitive to the cluster size and their inter-cluster distances. In our experiment, as 2-4 classes of image are assigned to each peer, having 3 signatures per peer makes the cluster denser and well separated as illustration Fig. 4.4. The variation of query performance against number of peers and TTL follows the characteristics described in Section 4.5.2 and 4.5.3. One might argue that having many signatures per peer should improve the query efficiency; however, this is just a trade off for the computation power used in clustering, maintaining and discovery of signatures and also it's corresponding attractive connection.

Table 4.7: Experiment parameters for data resolution experiments

Parameters	BFS	FQM-1	FQM-3
Number of Peer	2000 - 20000		
Topology follows power law distribution, with average branching degree	1.97	1.97	3.97
Number of documents assigned to each peer	200-400 documents 2-4 classes		
Signatures per peer	Nil	1	3
Dimension of extracted feature vector	9		
TTL value of the query packet	4,5,6,7,8,9		

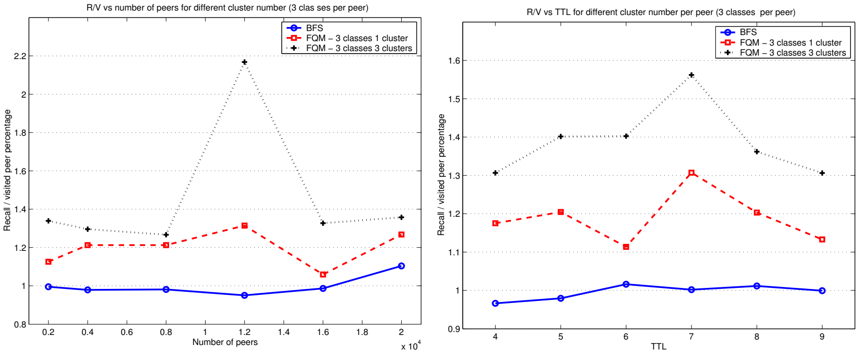


Figure 4.21: Query efficiency when different data resolution are used

4.5.5 Discussion

Here are a few more interesting points worthy of mentioning here. They are the local indexing method used, the effects of data distribution on performance, and the impact of number of attractive connections and TTL used on performance.

On the issue of local indexing used, there exists many high-dimensional spatial indexing methods used in content-based image retrieval systems, e.g., R-tree, VP-tree, etc., that can make the local search more efficient. However, the further elaboration of indexing methods is outside of the scope of the paper. In addition, most peers only share a limited number of images in the P2P network, a linear search shows acceptable response time in retrieving similar images using the DISCOVIR client program. More sophisticated methods can be added to the system later for improved performance.

In case some peers contain many images under different types of semantic meaning, there might be a concern for the query broadcasting problem since such kind of peers will probably maintain many attractive connections. To solve this problem, we restrict the maximum number of attractive connections made for a node to avoid the broadcasting condition. Specifically, we choose 3 as the maximum number used in the simulation experiments and implementation of DISCOVIR. Moreover, as the network grows, many clusters exist and there might be difficulties in reaching the cluster in a short number of hops. However, according to recent research, Gnutella network follows a power-law distribution, which has a special properties that diameter of the network is small [30]. A query message is guaranteed to reach a large portion of the network with TTL of 7. Referring to the query efficiency against TTL experiment as also shown in Fig. 4.19, TTL of 4 and 5 shows relatively low performance improvement, which illustrates the problem on not being able to reach the target cluster in a small number of hops. On the other hand, it shows that with TTL of 6 and 7, the system is able to attain a good performance in a network size of 10,000 peers.

Apart from those aforementioned, the point with utmost concern is whether FQM is applicable in real world P2P networks and how is its performanced affected under different data distributions. In addition to the Corel image dataset and synthetic dataset used in previous sections, we have added one more real world dataset to further evaluate FQM. We downloaded images from 18 different peers in the Gnutella network in July 2003. For each set of images shared by one peers, we assume that they are under the same category and form one cluster in the feature vector space. Using the statistics of inter-cluster distance, means, and variance, collected from real-world data, we generated another synthetic dataset to model the image distribution in real world P2P network. The characteristics of the four datasets used are listed in Table. 4.5.5 and Fig. 4.22. Indeed, even after modeling the real world data distribution, the FQM still shows an improvement.

Figure 4.23 shows the query efficiency of FQM under three different data distributions, with synthetic dataset being the most improved, Corel image dataset being the second and simulated real world dataset being the least improved. Nonetheless, FQM still obtains a gain of 20-30% in query efficiency when compared to BFS. Based on this observation, we conclude that the query efficiency of FQM is sensitive to data distribution. The best performance can be obtained if each peer shares one or small number of image categories and images in each category are closely clustered in the feature vector space. On the other hand, if peers share many different types of images, it is unlikely to form a community or cluster in the network, thus the performance would be the same as not having FQM at all. In conclusion, the query efficiency can be relate to the data distribution by the following relationship:

$$\textbf{Query Efficiency} \propto ID \times \frac{1}{VAR}, \tag{4.6}$$

where ID is the inter-cluster distance and VAR is the mean of variance of each class.

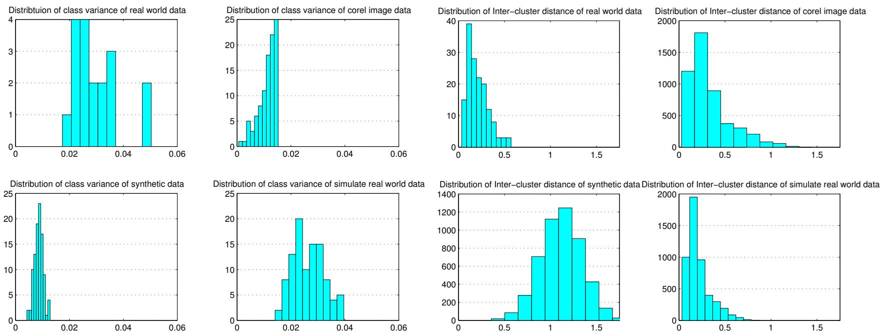
Table 4.8: Data distribution characteristics of 4 datasets

		Synthetic data	Corel image data	Real-world data	Synthetic data 2[a]
ID[b]	max	1.5297	1.4467	0.5718	0.8697
	min	0.3074	0.0272	0.0351	0.0318
	avg	0.9385	0.3298	0.2097	0.2138
VAR[c]	max	0.0388	0.0153	0.0504	0.0396
	min	0.0161	0.0006	0.0174	0.0141
	avg	0.0268	0.0113	0.0298	0.0263

[a] The synthetic data generated to simulate real-world data distribution.

[b] ID - inter-cluster distance between each classes.

[c] VAR - the mean of variance of each classes.

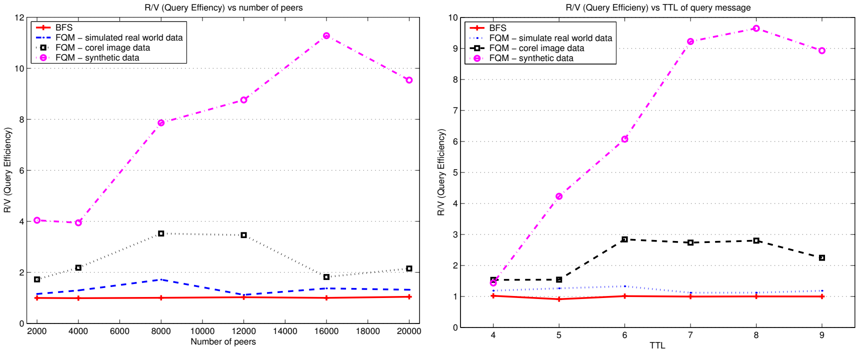


(a) (b)

Figure 4.22: (a) VAR distribution, (b) ID distribution of the four datasets.

4.6 Conclusion

In this chapter, we describe the implementation and architecture of a P2P system with CBIR functionality - DISCOVIR. With DISCOVIR, we can perform CBIR in a P2P network where peer shares, stores and index its own image collection. We propose a Peer Clustering and Firework Query Model to address the data location problem in P2P. It reduces network traffic and improves retrieval performance. We investigate the retrieval efficiency by simulations with different number of peers and TTL of query message. It shows a significant improvement when compared to BFS, a commonly used method in current P2P systems. We figure out the effect of different data distribution



(a) (b)

Figure 4.23: Query efficiency against (a) number of peers, (b) TTL of query message.

on this algorithm, which suggests ways to further improve query efficiency in real world situation.

Chapter 5

Future Works and Conclusion

In this thesis, two areas: relevance feedback and Peer-to-Peer architecture of CBIR are discussed. Relevance feedback intents to improve retrieval precision in a centralized architecture where the index of feature vectors are stored and indexed collectively. The Peer Clustering and Firework Query Model is a solution to the retrieval process when image collection and index of feature vectors are located distributively. Is relevance feedback applicable in a distributed environment? If yes, how? We left this question unsolved in this thesis. The underlying difficulty is that the feature indices are maintained distributively; thus successive queries overload the P2P network. If it is going to be applied in P2P, one of the possible ways might be using feedback information from user to infer which peer shares similar images apart from using signatures of peers. Another possible way might be building a user recommendation system in P2P. The relevance feedback information accumulated previously can be used to locate similar images in later queries. An architecture to propagate and store these feedback information in a distributed environment efficiently and unalterable by malicious peers is essential to such king of recommendation system.

In conclusion, we have explored two areas, learning image similarity and the need for databases, in the context of CBIR. In the aspect of learning image similarity, we proposed a parameters estimation and most informative display selection approach. This problem is comparable to estimating a distribution without making random sampling from the distribution. The proposed method combine closely the learning phase and display set selection phase to achieve a better estimation of parameters. This is because the display set selection impose a prior distribution on the samples (relevant images marked by user); thus, the learning process should account for this. Besides, we propose a SOM-based inter-query feedback learning technique to counterpoise that fact that clusters of real data are not Gaussian-like, which violates the assumption of parameters estimation approach in intra-query feedback. We demonstrated improvement in retrieval precision using synthetic and real world image data.

In the aspect of database support of CBIR, we demonstrate the possibility of using P2P systems as the source, the storage and the feature extraction computation for images. A self-organizing P2P network and query routing scheme are also proposed for making CBIR efficiently in P2P network. Experiments are done both on synthetic and real image data. Results show that the method proposed is scalable compared to existing Breadth First Search method in Gnutella. Experiments also reveal the effect of data distribution on query efficiency of our algorithm. We suggest that this is a trade-off between computation cost and query efficiency. Detail study of usage pattern in P2P network is need to determine the best number of signatures per peer. The methods proposed in this thesis open up the unexplored area of peer connection management and query routing of P2P systems when content-based similarity search is applied instead of exact match of filename.

Appendix A

Derivation of Update Equation

Expanding the expression stated in Eq. 3.4 and differentiate it w.r.t. δ_j assuming independcy among dimensions, we get:

$$\begin{aligned}
E &= \sum_{I_i \in I^+} \sum_{j=1}^{d} \Big(\frac{1}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} \Big) \times \Big(\frac{1}{\sqrt{2\pi}\delta_j} - \frac{1}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} \Big) \\
E &= \sum_{I_i \in I^+} \sum_{j=1}^{d} \Big(\frac{1}{2\pi\delta_j^2} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} - \frac{1}{2\pi\delta_j^2} \exp^{-\frac{(I_{ij}-\mu_j)^2}{\delta_j^2}} \Big) \\
\frac{dE}{d\delta_j} &= \sum_{I_i \in I^+} \Big(\frac{-1}{\pi\delta_j^3} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} + \frac{(I_{ij}-\mu_j)^2}{2\pi\delta_j^5} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}} \\
&\quad + \frac{1}{\pi\delta_j^3} \exp^{-\frac{(I_{ij}-\mu_j)^2}{\delta_j^2}} - \frac{(I_{ij}-\mu_j)^2}{\pi\delta_j^5} \exp^{-\frac{(I_{ij}-\mu_j)^2}{\delta_j^2}} \Big) \qquad \text{(A.1)}
\end{aligned}$$

We set $\frac{dE}{d\delta_j} = 0$, and subsitute $\frac{\delta_j^{\frac{1}{2}}}{\sqrt{2\pi}\delta_j} \exp^{-\frac{(I_{ij}-\mu_j)^2}{2\delta_j^2}}$ by $\delta_{old_j}^{\frac{1}{2}} P(I_{ij}|\theta_{old_j})$ in Eq A.1, after re-arranging terms, we get:

$$\begin{aligned}
\frac{dE}{d\delta_j} &= 0 \\
\delta_{new_j}^2 &= \frac{\sum_{I_i \in I^+}((I_{ij}-\mu_{old_j})^2 2^{\frac{1}{4}} \pi^{-\frac{3}{4}} - \delta_{old_j}^{\frac{1}{2}} P_{old}^{\frac{1}{2}} (I_{ij}-\mu_{old_j})^2 2^{\frac{1}{2}} \pi^{-\frac{1}{2}})}{\sum_{I_i \in I^+}(2^{\frac{1}{4}} \pi^{-\frac{3}{4}} - \delta_{old_j}^{\frac{1}{2}} P_{old}^{\frac{1}{2}} 2^{\frac{1}{2}} \pi^{-1})}
\end{aligned}$$

Here is the update equation for δ_j of each dimesnion.

Appendix B

An Efficient Discovery of Signatures

Consider a P2P network as shown in Fig. B.1. Suppose peer a wants to find out signatures (mean, standard deviation pair) of its neighboring peers within 3 hops, it sends out a signature query message to b, c, d first, then the message is propagated to $e, f, g, \ldots$ accordingly. When each peer receives this query message, it then replies it with its own signature(s) and IP address as shown in the left one. However, knowing the signatures of neighboring peers offers **too much** information in the decision of making attractive connections. According to Eq. 4.4, only the L_2 norm between means of two signatures is required to determine the content similarity of two peers. Base on this fact, we devise a more efficient signature discovery scheme with the following characteristics:

1. Each peer maintains its own signature and the signatures of all neighboring peers within 1 hop.

2. The signature query message contains signature(s) of the requester.

3. The signature reply message contains L_2 norm of means and IP address.

4. Each peer is responsible for calculation of distance between requester's signature and signatures of its neighboring peers within 1 hop.

Suppose all peers have already maintained the signatures of its neighboring peers within 1 hop. Peer a sends a signature query message to b, c, d containing its signature(s). Peer b receives this query, it calculates the distance between signature of peer a and peer e, f using Eq. 4.4 and reply the result together with IP address of e, f back to a. Peer b also forwards this message to e. Upon receiving this message, peer e repeats the procedure. It calculates the distance between signature of peer a and peer k, l and replies to a through b. All other peers follow the same method in answering the signature query and the process is similar to the right one of Fig. B.1. Finally, peer a gets a complete list of distance between its signatures and signatures from peer b to s. The algorithm is shown in Algorithm 4.

There are two possible ways for a peer to maintain signature(s) of neighboring peers within 1 hop. When a peer makes a signature query, it sends out its signature(s) in the message, thus neighboring peers capture the signature here. When a peer first joins the network, it exchanges its signature(s) with the connecting peer at the connection setup phase.

Algorithm 4 Algorithm for computing neighboring peers' signature efficiently

```
SignatureQuery(from-peer $f$, to-peer $v$, query $sig_r$, integer $ttl$)
for all $w \in Horizon(v, 1)$ except $f$ do
    for all $sig_w \in SIG_w$ do
        Compute $D(sig_r, sig_w)$ and reply to $f$
    end for
end for
if $ttl > 1$ then
    for all $w \in Horizon(v, 1)$ except $f$ do
        SignatureQuery($v$, $w$, $sig_r$, $ttl - 1$)
    end for
end if
```

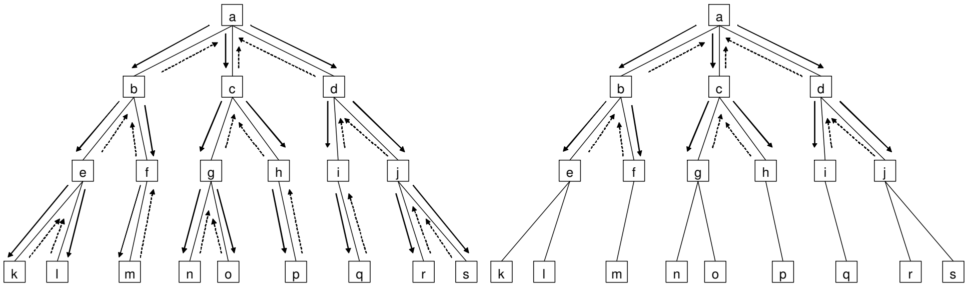


Figure B.1: Illustration of signature query message propagation:(left) original, (right) revised

Bibliography

[1] Gaurav Aggarwal, Ashwin T.V., and Sugata Ghosal. An Image Retrieval System With Automatic Query Modification. *IEEE Transactions on Multimedia*, 4(2):201–214, June 2002.

[2] M. R. Anderberg. Cluster Analysis for Applications. In *Academic Press, New York*, 1973.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, Addison Wesley, 1999.

[4] Farnoush Banaei-Kashani and Cyrus Shahabi. Criticality-based Analysis and Design of Unstructured Peer-to-Peer Networks as "Complex Systems". In *Proceedings of the Third International Workshop on Global and Peer-to-Peer Computing (G2P2PC)*, 2003.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedgins of ACM SIGMOD 1990*, pages 322–331, 1990.

[6] Michael Bergman. The deep web: Surfacing hidden value. Website, http://www.brightplanet.com/deepcontent/, September 2001.

[7] Chi-Hang Chan, Ka-Cheung Sia, and Irwin King. Utilizing Inter- and Intra-Query Relevance Feedback for Content-Based Image Retrieval. In *Proceedings of ICONIP'03, to be appeared*, Turkey, July 2003.

[8] Chia-Hui Chang and Ching-Chi Hsu. Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW. *IEEE Transactions on Konwledge and Data Engineering*, 11(4):595–609, July/August 1999.

[9] Wendy Chang, Gholamhosein Sheikholeslami, Jia Wang, and Aidong Zhang. Data Resource Selection in Distributed Visual Information Systems. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):926–946, November/December 1998.

[10] Yi-Shin Chen and Cyrus Shahabi. Yoda, an adaptive soft classification model: content-based similarity queries and beyond. *ACM/Springer Multimedia Systems Journal, Special Issue on Content-Based Retrieval*, 8:523–535, 2003.

[11] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems Concepts and Design*. Addison-Wesley, third edition, 2001.

[12] Ingemar J. Cox, Matt L. Miller, Thomas P. Minka, Thomas V. Papathomas, and Peter N. Yianilos. The Bayesian Image Retrieval System, PicHunter, Theory, Implementation, and Psychophysical Experiments. *IEEE Transactions on Image Processing*, 9(20-37), January 2000.

[13] Arturo Crespo and Hector Gracia-Molina. Routing Indices For Peer-to-Peer Systems. In *Proceedings of the Internation Conference on Distributed Computing Systems (ICDCS)*, July 2002.

[14] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis.

Journal of the American Society of Information Science, 41(6):391–407, 1990.

[15] A.P. Dempster, N.M. Laird, and Rubin D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.

[16] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, Niblack W., D. Petkovic, and W. Equitz. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, 3(3-4):231–262, 1994.

[17] The Freenet homepage. http://freenet.sourceforge.net.

[18] The Gnutella homepage. http://www.gnutella.com.

[19] Guo-Dong Guo, Anil K. Jain, Wei-Ying Ma, and Hong-Jiang Zhang. Learing Similarity Measure for Natural Image Retrieval With Relevance Feedback. *IEEE Transactions on Neural Networks*, 13(4):811–820, July 2002.

[20] Antonm Guttman. R-Trees - A Dynamic Index Structure for Spatial Searching. In *Proceedings of ACM SIGMOD 1984*, pages 47–57, 1984.

[21] Xiaofei He, Oliver King, Wei-Ying Ma, Mingjing Li, and Hong-Jiang Zhang. Learning a Semantic Space From User's Relevance Feedback for Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1), January 2003.

[22] Douglas R. Heisterkamp. Building a Latent Semantic Index of an Image Database from Patterns of Relevance Feedback. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)*, pages 134–137, August 2002.

[23] Irwin King and Zhong Jin. Relevance Feedback Content-Based Image Retrieval Using Query Distribution Estimation Based on Maximum Entropy Principle. In Liming Zhang and Fanji Gu, editors, *Proceedings to the International Conference on Neural Information Processing (ICONIP2001)*, volume 2, pages 699–704, Shanghai, China, November 14-18 2001. Fudan University, Fudan University Press.

[24] Irwin King, Cheuk Hang Ng, and Ka Cheung Sia. Distributed Content-Based Visual Information Retrieval System on Peer-to-Peer Networks. submitted to ACM TOIS, under revision.

[25] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, third edition, 2001.

[26] T. K. Lau and I. King. Montage: An image database for the fashion, clothing, and textile industry in Hong Kong. In *Proceedings of the Third Asian Conference on Computer Vision (ACCV'98)*, volume 1351 of *Lecture Notes in Computer Science*, pages I 410–417. Berlin, Germany: Springer Verlag, January 4-7, 1998.

[27] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[28] Joo-Hwee Lim, Jian Kang Wu, Sumeet Singh, and Desai Narasimhalu. Learing Similarity Matching in Multimedia Content-Based Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):846–850, September/October 2001.

[29] Modern peer-to-peer file-sharing over the internet. http://www.limewire.com/index.jsp/p2p.

[30] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and Replication in Unstructured Peer-to-Peer Network. In *Proceedings of 16th*

ACM International Conference on Supercomputing(ICS'02), New York, June 2002.

[31] S. Mehrotra, Y. Rui, M. Ortega, and T.S. Huang. Supporting Content-based Queries over Images in MARS. In *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pages 632–633, 1997.

[32] Multimedia Information Processing Lab - CUHK. http://www.cse.cuhk.edu.hk/∼miplab.

[33] The Napster homepage. http://www.napster.com.

[34] A. Natsev, R. Rastogi, and K. Shim. WALRUS: A Similarity Retrieval Algorithm for Image Databases. In *Proc. SIGMOD, Philadelphia, PA*, 1999.

[35] Cheuck Hang Ng, Ka Cheung Sia, and Chi Hang Chan. Advanced Peer Clustering and Firework Query Model. In *Poster Proceedings of The Twelfth International World Wide Web Conference, Poster ID S130*, Hungary, May 2003.

[36] Cheuk Hang Ng and Ka Cheung Sia. Peer Clustering and Firework Query Model. In *Poster Proc. of The 11th International World Wide Web Conference*, May 2002.

[37] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM, 1999.

[38] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: tools for content-based manipulation of image databases. In *Proc. SPIE*, volume 2185, pages 34–47, February 1994.

[39] S. Ratnasamy, P. Francis, M. Handley, and S. Karp R. Shenker. A Scalable Content-Addressable Network. In *In Proc. ACM SIGCOMM*, August 2001.

[40] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proceedigns of the International Peer-to-Peer Workshop*, 2002.

[41] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[42] 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*, 10:39–62, April 1999.

[43] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrota. Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval. *IEEE Transactions on Circuits and Video Technology*, 8(644-655), September 1998.

[44] D.E. Rumelhart and D. Zipser. Feature Discovery by Competitive Learning. In *Cognitive Science*, 1985.

[45] The Search for Extraterrestrial Intelligence homepage. http://www.setiathome.ssl.berkeley.edu/.

[46] K. C. Sia and Irwin King. Relevance Feedback Based on Parameter Estimation of Target Distribution. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages 1974–1979, May 2002.

[47] Ka Cheung Sia, Cheuk Hang Ng, Chi Hang Chan, Siu Kong Chan, and Lai Yi Ho. Bridging the P2P and WWW Divide with DISCOVIR - DIStributed COntent-based Visual Infomration Retrieval. In *Poster Proceedings of The Twelfth International World Wide Web Conference, Poster ID S172*, Hungary, May 2003.

[48] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jai. Content-Based Image Retrieval at the End of the Early Years. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[49] J. R. Smith and S. F. Chang. An Image and Video Search Engine for the World-Wide Web. In *Proc. SPIE*, volume 3022, pages 84–95, 1997.

[50] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient Contnet Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of Infocom'03*, 2003.

[51] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, August 2001.

[52] Chuqiang Tang, Zhichen Xu, and Mallik Mahalingam. PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks. Technical report, HP-Lab, October 2002.

[53] Q. Tian, P. Hong, and T.S. Huang. Update Relevant Image Weights for Content-Based Image Retrieval Using Support Vector Machines. In *Proceedings to the IEEE Conference on Multimedia and Expo*, volume 2, pages 1199–1202, June 2000.

[54] Ashwin T.V., Rahul Gupta, and Sugata Ghosal. Adaptable Similarity Search using Non-Relevant Information. In *Proceeding of the 28th VLDB Conference*, August 2002.

[55] J. Z. Wang, G. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. In *IEEE Trans. on pattern Analysis and Machine Intelligence*, volume 23, pages 947–963, 2001.

[56] Y. Wu, Q. Tian, and T.S. Huang. Discriminant-EM Algorithm with Application to Image Retrieval. In *Proceedings to the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 222–227, June 2000.

[57] Cheng Yang. Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity. In *Proceedings of ACM Multimedia*, 2002.

[58] Cheng Yang. Peer-to-Peer Architecture for Content-Based Music Retrieval on Acoustic Data. In *Proceedings of the twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[59] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Computer Science Division, U.C. Berkeley, April 2001.